NUMERICAL MODELLING OF FULLY COUPLED VORTEX-INDUCED VIBRATION OF ELASTICALLY MOUNTED 3D OFFSHORE SPAR PLATFORM

A thesis paper by

ANANYA MAJUMDAR
EXAMINATION ROLL NO. M4CIV23015
REGISTRATION NO. 160025 of 2021-2022

Under the guidance of

PROF. Dr. PARTHA BHATTACHARYA

Submitted in the partial fulfilment of the requirements for the degree of

MASTER OF ENGINEERING IN CIVIL ENGINEERING (STRUCTURAL ENGINEERING)

DEPARTMENT OF CIVIL ENGINEERING
FACULTY COUNCIL OF ENGINEERING & TECHNOLOGY
JADAVPUR UNIVERSITY
KOLKATA – 700032, INDIA

JUNE-2023

Department of Civil Engineering

Faculty Council of Engineering & Technology

JADAVPUR UNIVERSITY

KOLKATA – 700 032, INDIA

1. Title of the thesis: **Numerical Modelling of Fully Coupled Vortex-Induced Vibration of Elastically Mounted 3D Offshore Spar Platform**

2. Name, Designation & Institution of the Supervisor/s:

1. Dr. Partha Bhattacharya

Head of Department

Department of Civil Engineering

Jadavpur University,

Kolkata 700032, India

3. List of publication:

Journal Publications

[1] Adhikary, B. R., **Majumdar, A.**, Sahu, A., & Bhattacharya, P. (2023). Sensitivity of TBL Wall-Pressure over the Flat Plate on Numerical Turbulence Model Parameter Variations. *CFD Letters*, *15*(7), 148-174. https://doi.org/10.37934/cfdl.15.7.148174

[2] Adhikary, B. R., **Majumdar, A.**, Sahu, A., & Bhattacharya, P., Finite Element Modelling of transmission of TBL induced vibrational energy transfer through a stiffened double panel using CFD sensitivity study. (Under review)

[3] **Majumdar, A.**, Adhikary, B. R., & Bhattacharya, P., A fully coupled vortex-induced vibration (VIV) model for offshore spar platforms. (Under preparation)

4. List of Patents – Nil

5. List of Presentations in National / International Conference:

[1] **Majumdar, A.**, Adhikary, B. R., & Bhattacharya, P., Reduction in turbulence-induced non-linear dynamic vibration using tuned liquid damper (TLD), *Proceedings of the 9th International and 49th National Conference on Fluid Mechanics and Fluid Power (FMFP)*, Springer, December 14-16, 2022, IIT Roorkee, India.

[2] B. R. Adhikary, **A. Majumdar**, S. Sarkar, P. Bhattacharya, Sensitivity mapping between different TBL wall- pressure spectra and CFD models using OpenFOAM and Fluent solver, *Proceedings of the 9th International and 49th National Conference on Fluid Mechanics and Fluid Power (FMFP)*, Springer, December 14-16, 2022, IIT Roorkee, India.

DEPARTMENT OF CIVIL ENGINEERING

FACULTY OF ENGINEERING AND TECHNOLOGY

JADAVPUR UNIVERSITY

KOLKATA – 700032

CERTIFICATE OF RECOMMENDATION

This is to certify that the thesis entitled "**Numerical Modelling of Fully Coupled Vortex-Induced Vibration of Elastically Mounted 3D Offshore Spar Platform**", submitted by **Ananya Majumdar** (Examination Roll No. M4CIV23015, Registration No. of 160025 of 2021-2022) of Jadavpur University is absolutely based upon her own work under the supervision of **Dr. Partha Bhattacharya.** I hereby recommend that the thesis be accepted in partial fulfilment of the requirements for awarding the degree of '**Master of Engineering in Civil Engineering (Structural Engineering)**'.

Countersigned by

Dr. Partha Bhattacharya	**Dean, FET**
(Professor)	**Jadavpur University**
Head of Department	**Kolkata 700032**
Department of Civil Engineering	
Jadavpur University, Kolkata 700032	
India	

DEPARTMENT OF CIVIL ENGINEERING

FACULTY OF ENGINEERING AND TECHNOLOGY

JADAVPUR UNIVERSITY

KOLKATA – 700032

CERTIFICATE OF APPROVAL

This thesis paper is hereby approved as a credible study of an engineering subject carried out and presented in a manner satisfactorily to warrant its acceptance as a pre-requisite for the degree for which it has been submitted. It is understood that, by this approval the undersigned do not necessarily endorse or approve any statement mad, opinion expressed or conclusion drawn therein, but approves the thesis paper only for the purpose of which it was submitted.

Committee of Thesis Paper Examiners

_____

Signature of Examiner

DECLARATION

I, Ananya Majumdar, Master of Engineering, in Civil Engineering (Structural Engineering), Faculty of Engineering and Technology, Jadavpur University, hereby declare that the work being presented in the thesis titled **"Numerical Modelling of Fully Coupled Vortex-Induced Vibration of Elastically Mounted 3D Offshore Spar Platform"**, is an authentic record of work that has been carried out in the Department of Civil Engineering, Jadavpur University, Kolkata under the guidance of **Dr. PARTHA BHATTACHARYA**, Professor, Head of Department, Department of Civil Engineering, Jadavpur University, Kolkata. This work has not previously been submitted for a degree or diploma in any University. To the best of my knowledge and belief, the thesis contains no material previously published or written by another person except where due reference is made in the thesis itself.

Place: Kolkata

Date:

_____

Ananya Majumdar

Examination Roll No. M4CIV23015

Registration Number: 160025 of 2021-2022

Class Roll No.: 002110402001

ACKNOWLEDGEMENTS

Foremost, I would like to express my deepest gratitude to my advisor Prof. Dr. Partha Bhattacharya, Head of Department, Department of Civil Engineering, Jadavpur University, Kolkata, for providing constant guidance, independence to work on Fluid-Structure Interaction with every possible resource and close supervision in all respect throughout.

I would like to specially thank Mr. Biplab Ranjan Adhikary, Research Scholar, Department of Civil Engineering, Jadavpur University, Kolkata, without whose active support in the form of teaching, continuous guidance, extremely valuable suggestions, trouble-shooting and encouragement in ups and downs throughout the entire research period, this piece of work would have been impossible to be delivered.

I am thankful to my parents and sibling for the constant support in every form, without which none of this would have been possible.

I would also like to extend my thanks to all the faculties of Department of Civil Engineering, Jadavpur University for their valuable inputs during Term paper leading to thesis presentation.

Date:

_____

Ananya Majumdar

Examination Roll No. M4CIV23015

Registration Number: 160025 of 2021-2022

Class Roll No.: 002110402001

ABSTRACT

This thesis presents a comprehensive investigation into the numerical analysis of canonical flow problems and the subsequent development of a vortex-induced vibration (VIV) model for offshore spar platforms. The study utilizes in-house MATLAB coding and the OpenFOAM computational fluid dynamics (CFD) software, incorporating turbulence modelling techniques such as Reynolds-averaged Navier-Stokes (RANS) and Large Eddy Simulation (LES) to address the complexities associated with turbulent flows.

In the initial part of the thesis, a systematic analysis of canonical flow problems is conducted to establish a solid foundation for understanding fluid dynamics phenomena. Several canonical flow scenarios, including lid-driven cavity, and flow past a static cylinder, are considered. The governing fluid dynamics equations are solved using a meticulously developed in-house MATLAB code, which employs appropriate numerical schemes and boundary conditions. The accuracy and reliability of the code are validated by comparing the obtained results with existing analytical solutions and experimental data. This is further extended to an in-house VIV model with an elastically mounted cylinder experiencing the fluid flow.

The second segment of the thesis focuses on addressing the vortex-induced vibration phenomenon prevalent in offshore spar platforms. VIV poses a significant concern for these structures, as the excessive vibration can lead to fatigue damage and structural failure. Mooring lines and other attachments can also get damaged in case of tethered floating structures. In order to simulate and predict VIV behaviour accurately, a specialized 3D fully-coupled VIV model is developed utilizing the C++ based open-source flow solver OpenFOAM. The model incorporates turbulence modelling techniques, including RANS and LES, to capture the complex turbulent flow features and their impact on VIV. This allows for a more realistic representation of the flow characteristics around the offshore spar platform. The RANS is used in the case of two-way coupled model, whereas, LES is employed for one-way coupled vibration model.

The developed VIV model with turbulence modelling is meticulously validated against available experimental data. The validation process involves a comprehensive comparison of predicted VIV response amplitudes, RMS (root mean square) lift coefficient for different reduced velocities, with the corresponding measured values, ensuring the accuracy and reliability of the model. The successful validation of the VIV model with turbulence modelling confirms its capability to accurately predict the VIV behaviour of offshore spar platforms under turbulent flow conditions. Furthermore, the VIV study extended to a multi degrees of freedom system, and effect of the spring stiffness on VIV response amplitude, frequency, etc. are studied.

The thesis concludes by providing a detailed discussion on the findings, limitations, and potential areas for future research. The inclusion of turbulence modelling using RANS and LES in the VIV model enhances the accuracy of predicting the VIV characteristics in turbulent flow environments. Furthermore, the numerical results obtained from solving canonical flow problems using the in-

house MATLAB code and incorporating turbulence modelling techniques demonstrate the effectiveness of the approach in accurately simulating and analysing a wide range of fluid dynamics phenomena.

Keywords: Computational fluid dynamics, canonical flow problems, vortex-induced vibration, offshore spar platform, MATLAB coding, OpenFOAM, turbulence modelling, RANS, LES, fluid-structure interaction.

CONTENTS

[This page is intentionally left blank]

CHAPTER 1: INTRODUCTION

1.1 BACKGROUND

Civil engineering structures are exposed to many types of forces, such as dead load, live load, seismic load, fluidic forces and so on. These external forces can cause detrimental effects on the structures, like collapse, fatigue, vibration, deflection, etc. The standard codes are generally used to design and analyse the structures to be built. Taller structures, however, provide additional challenges in terms of structural design because their dynamic response to wind induced excitation becomes more visible [1]. This is why, for the final design, often a wind-tunnel test is carried out. The wind tunnel test employs fluid dynamics to ensure the building's safety in an occurrence of severe weather. In case of tall buildings exposed to high-speed wind, the wind tunnel test fetches the data from which the stability of the building can be understood. In order to attenuate the force-excited structural motion, various active and passive dampers are used. Similarly, civil engineering structures like dams, hydraulic channel, offshore structures etc. are exposed to forces of flowing fluids. However, solid mechanics approaches may not be enough for analysing the dam's sensitivity to water pressure. Fluid mechanics is crucial and has become an ancillary science in the advancement of technology in engineering and can substantially aid in the growth and advancement of numerous disciplines.

Wind tunnel tests or laboratory experiments of models are not always very feasible options. So, instead of these, computational fluid dynamics (CFD) is being used to replicate the wind-tunnel test or experiments numerically to both make the process economical and less time consuming. Civil engineers may use computational fluid dynamics (CFD) to inspect the hydraulic condition and perform the appropriate treatments or repairs, when complicated problems in any hydraulic structure arise as a result of rapid change or unanticipated deterioration [2]. Multiple modelling methods are utilised as an inexpensive means to build reliable offshore wind turbines to mitigate wind damage.

Apart from these, CFD simulations are used in Environmental engineering, Geotechnical engineering, Hydraulic engineering, Materials science, Structural engineering, Transportation engineering, and Wind engineering.

- CFD can be used in *Geotechnical Engineering* for flow across granular dam filters, simulation of particle migration, and finite element analysis on the weak foundation of an oil storage tank.
- CFD can be used for *structural analysis*, predicting vibration for pumping platforms, and assisting the oil and gas industry by increasing the longevity and safety of drilling derricks and substructures through stress analysis, two-way interaction between thermal solid elements and

structural beam and shell elements, and non-linear analysis of shear dominant pre-stressed concrete beams.

- In *Transportation Engineering*, CFD can be used to bridge the gap in loading capacity, simulate temporary bridges, aid in disaster relief, ventilate enormous railway tunnels, improve subterranean safety and ventilate metro trains.
- CFD can be used in *Environmental Engineering* to capture natural energy. Using many simulation tools to build reliable offshore wind turbines, combining solar panels with roof profiles - simulation alerts the design of new solar panel frames, saving moulding time, material, and cost. CFD can be used to anticipate blocking areas in a building's drainage system.
- Computational fluid dynamics techniques can be used to simulate flow parameters like friction losses and local head losses in *irrigation engineering*.

From the above applications of fluid dynamics and thus computational fluid dynamics in civil engineering, the phenomena that should be taken care of at most is fluid-induced vibration. Some of the most important fluid flow phenomena are:

- A *boundary layer* is a thin layer of fluid created by the fluid flowing along the surface of a bounding surface. The contact of the fluid with the wall causes a no-slip boundary condition (zero velocity at the wall). The flow velocity then grows monotonically beyond the surface until it reaches the bulk flow velocity. The velocity boundary layer is a tiny layer of fluid whose velocity is yet to be reverted to the main flow velocity.
- A *convection* cell is a phenomenon that happens in the realm of fluid dynamics when density variations exist within a body of liquid or gas. Convection is the movement of liquids, and the moving body is referred to as a convection cell.
- *Drag* is a kind of resistive force in fluid dynamics that acts in the opposing direction of any object moving with regard to a surrounding fluid. This can occur across two fluid layers or surfaces, as well as between a fluid and a solid surface.
- *Hydrodynamic stability* is the study of fluid flow stability and the start of instability. The scientific investigation of hydrodynamic stability seeks to determine whether a given flow is stable or unstable, and if so, how these instabilities will cause turbulence to form.
- A fluid flowing around an object applies pressure to it whose component perpendicular to the direction of the incoming flow is called *lift*. In contrast, the drag force is the component of the force that is parallel to the flow direction. Lift is typically applied in an upward direction to counteract the force of gravity, but it can be applied in any direction at right angles to the flow.
- *Vortices* are defined by the distribution of velocity, vorticity (the curl of the flow velocity), and the idea of circulation. Turbulent flow is dominated by vortices. The fluid flow velocity in most vortices is greatest near the axis and decreases in inverse proportion to distance from the axis.

All of these can arise in ‘controlled aerodynamic instability’ phenomena, which was first used by Cristiano Augusto Trein in the Nineteenth KKCNN Symposium on Civil Engineering held in Kyoto – Japan in 2006. These phenomena include Kármán vortex street, flutter, galloping and buffeting.

- A *Kármán vortex street* is a recurring pattern of whirling vortices created by vortex shedding, which is responsible for the unstable separation of flow around blunt things in fluid dynamics. It is generally formed at low Reynolds number.
- *Flutter* is a dynamic instability of an elastic structure in a fluid flow generated by positive feedback between body deflection and fluid flow force. The "flutter point" in a linear system is the point at which the structure is undergoing simple harmonic motion, zero net damping and any further drop in net damping leads to self-oscillation and eventual breakdown.
- *Galloping* is a fluid-induced, low-frequency, large-amplitude oscillations. According to the galloping instability analysis, the incidence of negative aerodynamic damping is largely dependent on the critical flow condition.
- The *lock-in phenomena* happens when the structure oscillates at a frequency close to the stationary structure’s natural vortex shedding frequency, and the vortex shedding frequency of the wake flow remains in line with the cylinder's vibration frequency.
- *Buffeting* is a high-frequency instability generated by separation of airflow or shock wave oscillations resulting from one object colliding with another. It occurs by a rapid increase in load and thus is a forced random vibration.

Thus, it can be concluded that fluid flowing past structures can lead to vortex formation, which in turn can cause the structure to vibrate. Such vibrations due to vortex shedding is known as Vortex-induced vibrations (VIV). This is the why the vortex-induced vibrations of rigid structures is an important area of research.

These vibrations can be significant and prove to be fatal for the structure when the shedding frequency is near the natural frequency of the structure. VIV can occur at low as well as high Reynold’s number regime.

Some examples of civil engineering structures prone to vortex-induced vibrations are tall buildings, flue-gas stacks, or industrial chimneys, launch vehicles, transmission lines, submarine periscope and other offshore structures. It can cause large-amplitude vibrations of tethered structures in the ocean. These are tall and slender structures with circular cross-sections having large diameters. Chimneys are designed accurately to make them economic and safe so that they can withstand both static and dynamic forces. The forces can be static wind load, turbulence, or vortex shedding. So, the static deformation and dynamic amplitude response are to be predicted accurately. This can further help in reducing the large amplitude response by designing effective active or passive dampers that can reduce the response. However, the aerodynamic forces on a

circular cylinder are highly dependent on Reynolds number, and thus it is difficult to recreate or replicate the large structures in the field, both experimentally and numerically, to predict the accurate response. There are many predictive models in IS codes or scientific literature but many open questions remain on how to best include all aerodynamic and structural parameters. These models may not be able to take the effects of turbulence, surface roughness, or any active or passive damper.

In offshore oil and gas engineering, the VIV of cylindrical structures is an important design issue. Due to increasing demand for crude oil, offshore gas, and oil; explorations have been moved to deeper water levels. Offshore floating wind turbines are used to conserve energy and generate electricity. It can further help to reduce visual pollution, and achieve stronger and more constant winds. VIV can cause large-amplitude vibrations of tethered structures in the ocean. Thus, the effect of VIV on floating structures needs to be studied.

1.2 MOTIVATION OF THE PRESENT RESEARCH

The recent global trend of the 'Green Shift' from fossil energy (coal, oil, and natural gas) to renewable energy (sunlight, wind, waves, and tide) is the most important aspect in order to mitigate the problems that come with development [3]. DNV forecasts that by 2050, the installed floating global wind capacity will grow from today's 100 megawatts (MW) to over 264 gigawatts (GW) [4], [5] so that 2% of the world's electricity demand can be supplied by the cost-efficient and dependable floating offshore wind. Floating offshore wind turbines are considered a viable solution in ocean depths of more than 50 to 60 metres and with abundant wind resources. As a rapidly growing technology, it has the ability to use less foundation material, reduce the installation cycle and decommissioning, and generate more wind power. New risks can be managed by using off-the-shelf bottom fixed turbines and well-known oil and gas technology for floaters.

The different types of offshore floating wind turbines are shown in Fig 1.2. It can be seen from Fig. 1.2, that the most common structure used for the extraction of oil and gas from the deep sea is that of a truncated cylindrical floating structure.

Thus, the vortex-induced vibrations of a truncated circular cylinder are computed numerically in this thesis work. To obtain this, at first some canonical flow problems have been simulated by in-house MATLAB coding to get a strong understanding of the flow physics for flow past bluff bodies.

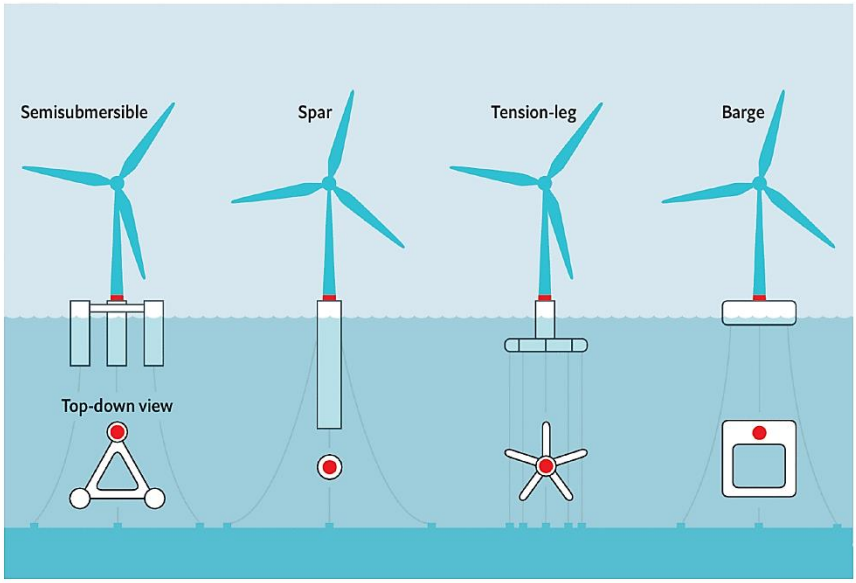


Fig. 1.2: Various types of offshore floating wind turbine depicting the most common cross-section involved [6]

1.3 ORGANIZATION OF THE DISSERTATION

The present dissertation contains six chapters. The present chapter gives a general introduction to vortex-induced vibrations of floating offshore structures. Different aspects of vortex-induced vibration in civil engineering structures are also discussed in the context of the present research. The basic motivation of the work is thus derived and is mentioned herein.

A detailed review of the previous research activities involving vortex-induced vibrations of cylindrical structures and control strategies of the vibration is presented in Chapter 2. The thesis has been broadly classed into six sections:

In chapter – 1, the exposition of real-life civil engineering structures to various fluidic force is explained. Thus, the motivation of the present work is obtained herewith.

In chapter – 2, available literatures on vortex-induced vibrations and development of Reynolds Averaged Navier-Stokes (RANS) turbulence models over time are mentioned and the gaps from the literature which have been the main objectives of the present thesis are discussed.

In chapter – 3, mathematical basis to solve the Navier-Stokes equation and thus address the two-way coupling through fluid-structure interaction is detailed. The entire finite volume methodology used to solve the equation along with various schemes used in the present work is explained. The methodology used to in-house code, solve various cases of flow past bluff bodies are explained here.

In chapter – 4, the numerical methodology used to solve the entire vortex-induced vibration of cylinder is detailed. The application of those in OpenFOAM is also shown.

In chapter – 5, numerical results obtained from the various works done is shown.

In chapter – 6, the final chapter, conclusion of the present study has been drawn and future scope of research work has been mentioned.

The scope of the current work has been determined and the current research objectives are formed through a critical analysis of previous works. The steps required to achieve the goals are also outlined.

CHAPTER 2: LITERATURE REVIEW

2.1 INTRODUCTION

Various applications of fluid dynamics in civil engineering structures have been explained in Section 1.1 of Chapter-1. These applications are detailed in [7] and [8]. Because of its significant impact on engineering, vortex-induced vibration of cylinder has been widely explored around the world during the last few decades. Several reviews regarding experimental and numerical investigation for VIV in cylindrical structures have been published, some of which are in [9]. It is specified in [9] that for sharp-edged bluff bodies the point of separation is fixed, whereas for bluff bodies with continuous edges the location of separation depends on the shape of the body and the state of the boundary layer. This forms the basis of switching from in-house coding of flow past square cylinder to the flow past circular cylinder in the present work. The important parts of the evolution of concepts, theoretical insights, experimental methodologies, and numerical models are documented systematically in [10]. The vortex dynamics and energy transfer that cause modes of vibration, as well as the importance of mass and damping, the concept of a critical mass, the relationship between force and vorticity, and the concept of "effective elasticity," are specifically mentioned in [11]. The lock-in phenomena and effect of mass and damping of body in VIV is explained in [12], along with which it is also remarked that controlling VIV is quite difficult. The vibration pattern of bluff bodies, impact of fluid force on structure are discussed in [13]. Similar reviews have been done in [14], [15], [16], [17], [18]. There are many studies on vortex-induced vibrations of an elastically mounted cylinder. As the vortex shedding frequency synchronizes with the vibration frequency, a high amplitude response is obtained which indicates that the Strouhal law[1] is not followed by vortex shedding frequency. The lock-in phenomena happens when the cylinder oscillates at a frequency close to the stationary cylinder's natural vortex shedding frequency, and the vortex shedding frequency of the wake flow remains in line with the cylinder's vibration frequency. This lock-in phenomenon was found to occur in a wide range of reduced velocity (V_r) [19], which is described in Eq 5.1, Chapter-5. The lock-in range of reduced velocity is dependent on the mass ratio[2], m*, [20].

[1] A universal relationship between the frequency of oscillation induced in a restrained fluid body by an external free flow, the velocity of the latter, and a typical width of the body, is established.

[2] Mass ratio

$$m^* = \frac{Mass\ of\ cylinder}{Mass\ of\ displaced\ fluid}$$

2.2 TYPES OF VORTICES AS PER LITERATURE

As the oscillation amplitude increases, the range of frequency ratios over which this "lock-in" of frequencies are possible increases [21]. It can lead to structural fatigue and instability under certain circumstances. The vortex synchronization regions in the wavelength-amplitude plane have been studied. The process of pairing of vortices is a necessary base for repeatable vortex-shedding pattern and for enabling vortices to convect away from the local body region under their own induced velocities. A vortex pair is denoted by 'P', and a single vortex is denoted by 'S'. A pattern where in each cycle a vortex pair and a single vortex are shed is known as 'P+S'. The designation 2S means that in each half cycle, a vortex is fed into the downstream wake, like the natural Karman vortex shedding; 2P means the formation of vortex pairs which convect laterally outwards from the wake centerline, and the P + S mode is an asymmetric version of the 2P mode where the cylinder sheds a pair and a single vortex each cycle. Mode 2P* is similar to 2P except that vortex pairs in one of the half cycles convect away from in front of the body. In this case, the convection of each pair is in the downstream direction (creating a jet), rather than in the upstream direction for the 2P mode.

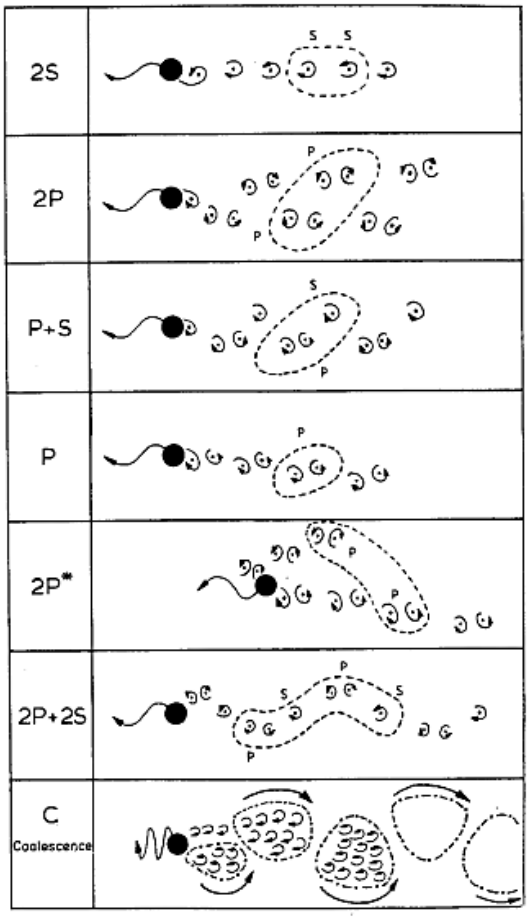


Fig. 2.2: Sketches of vortex shedding patterns in Fig. 3(b) of [21]

A critical curve in the amplitude-wavelength plane is defined as, where the resonant synchronization, mode transition occurs, that is, transition from one mode of vortex formation to another[21]. This is shown in Fig. 4 of [21].

2.3 VIV OF ELASTICALLY MOUNTED CYLINDERS

From a free-vibration study, using the Digital particle image velocimetry (DPIV) technique for vorticity measurements, two types of response are obtained, depending on the mass-damping parameter ($m * \xi$), where 'm' is the mass of the structure and ξ is the damping ratio [22]. In the high ($m * \xi$) case, there occur two response amplitude modes, which are called the 'initial' and 'lower' branches [22] [23]. As reduced velocity increases, a jump in amplitude and, in phase angle between force and displacement is observed, across the mode transition. For low ($m * \xi$) case, there exists three branches, the third one being 'upper' branch in which the vortex wake is also in the 2P mode [24], [25]. Another branch named the 'super upper branch' occurs when the structure is allowed to vibrate in both inline and cross-flow directions, where the peak-to-peak amplitude in the cross-flow direction is increased up to 3 times in diameter [26]. The vortex shedding in this branch is found in '2T' mode, which means two triplets of vortices were shed from the structure in each cycle of vibration.

There have been many numerical studies to investigate VIV of cylinders in fluid flow. The responses in the initial and lower branches for single degree of freedom VIV in cross-flow direction are predicted well by numerical models based on Reynolds-Averaged Navier-Stokes (RANS) [27], [28]. But there is a place of dissatisfaction with respect to the response in the upper branch. Satisfactory results in all branches were obtained for 2DOF VIV of circular cylinder in steady flow using 2D RANS [29]. Numerical studies on 2D Navier-Stokes equation for VIV of single cylinder at low Reynolds number (between 100 and 200) have been done in [30]–[33]. A finite-element study of two degrees of freedom oscillations of circular cylinder at low Reynolds number was carried out in [31]. The hysteresis and vortex shedding modes have been shown in [32]. The flow interference between a stationary and an elastically mounted cylinder in proximity is studied in [34]. Similar study for two rigidly coupled circular cylinders in tandem and side-by-side arrangements at a Reynolds number of 150 is done in [35]. The effect of cubic stiffness nonlinearity on the vortex-induced vibration of a circular cylinder at low Reynolds numbers has been studied in [36]. Thus, the studies for tandem cylinders have been done in [34]–[36]. In most of the engineering applications, the wake behind a 2D cylinder for uniform flow is 3D in nature. The three dimensionality is relatively weak in the near wake region of 2D cylinders as the vortex shedding is parallel to the cylinder axis and regular. So, previously due to computational limitations, 2D numerical models were used to simulate flow past 2D cylinders. As the flow around a cylinder is completely three-dimensional even at low numbers, hence 3D simulations for flow past cylinder has been carried out in this study. Simulation of VIV of a cylinder at low Reynolds

number in the turbulent flow regime by solving the 3D Navier Stokes equation with a Finite Element approach (Petrov-Galerkin FEM) has been carried out earlier [20], [37]–[41]. The vortex mode selection of a rigid cylinder exposed to VIV at low mass-damping has been studied in [38]. The free vibrations of a 3D cylinder at Reynolds number 1000 are studied in [39] and its intermittency in free vibration is studied in [40]. The same has been done for two-degrees-of-freedom VIV of cylinder in [41].

Most of these studies have neglected the effect of free end by the fixation of a flat plate at the free end of the cylinder of finite length to prevent fluid from flowing around the plate. The formation of the vortex streets was suppressed at the free-ends and the approaching flows were boundary layer flows. The turbulent uniform flow around cylinders of finite length have been studied in [42]. Besides this, the wake flow near the free-end was dominated by the downwash and the trailing vortices [43]. The study of uniform flow past one or two circular cylinders is done in [44]. The effect of free end on the near wake of the flow structure has been studied in [45] and the flow over the free end surface of the same has been studied in [46]. The tip vortex structure for the same was founded in [47]. The aerodynamic loads and wake flow features of low aspect-ratio triangular prisms at different wind directions were experimentally investigated in [48].

One of the most important and basic elements on which VIV of cylinders depend is the aspect ratio. The effect of free end is also a matter to be taken care of. It has a significant effect on the wake of the cylinder and vortex formation. The shear layer separated from the free end of a finite cylinder has significant effect on wake pattern. This has been studied thoroughly experimentally and numerically for low aspect ratio[3] of cylinders. Large Eddy simulations and experiments for flow around cylinders of finite length were done in [49]. A review on the effect of free-end in the cylinder wake is done in [50]. Similar study has been done in [51]. The effects of free end of a cylinder with a base column have been studied in [52]. At the same time, the cylinder-base concept is being used in some floating structure designs due to its benefits in stability and heave damping, such as the DeepCwind semi-submersible platform created to support the OC4 floating offshore wind turbine (FOWT) [53]. Some researchers have found out that the flow results for VIV of a cylinder changes for different aspect ratios. Further studies on VIV of offshore cylindrical structures have been studied [54]–[59]. The study of transverse vibrations of an elastically mounted cylinder exposed to an oscillating flow was done by [58] and a span wise correlation on the same was found by [59] Numerical simulation of VIV of four circular cylinders in a square configuration was studied in [56]. Two degrees-of-freedom VIV of circular cylinder is also studied numerically in [57].

[3] Aspect Ratio is the ratio of axial length to diameter for cylindrical structures with a circular cross-section

Both ends of finite-length rigid circular cylinders were exposed to the flow, in the wind tunnel experiments, which discovered a 200% wider lock-in regime along with a 230% rise in peak response amplitude as L=D was reduced from 28.8 to 5 [60]. The VIV of offshore cylindrical structures due to ocean waves are studied in few of these works.

2.3.1 Gap from Literature: From these literature study, it can be seen that the effect of spring arrangement, stiffness or number of springs used is not explicitly specified or studied. Also, the effects of free-end on a floating 2DOF VIV of cylinder at the same stiffness of the system as that of the 1DOF case, thus giving a sense of releasing degrees-of-freedom, has not been studied yet. [The single degrees-of-freedom system considered in Section-5.3 is abbreviated as '1DOF' and for the multi degrees-of-freedom system is abbreviated as '2DOF']. This is similar to the releasing of one or two of the six restraints in a rigid body, keeping the stiffness of the system same.

2.4 REYNOLDS AVERAGED NAVIER STOKES EQUATION

As in the present work, the Reynolds Averaged Navier-Stokes equations hold a very vital role, so its development and closure of the equation with the help of various turbulence models is explained in this section.

The Reynolds stresses are the ones that are to be computed, to find a solution to the RANS Equation. To solve this, Boussinesq made an assumption in 1877 known as the Boussinesq hypothesis or Eddy-viscosity model (EVM). In this hypothesis, turbulence, which are in the form of eddies, are considered as particles in Brownian motion, although turbulence is not random in nature. But turbulence is caused by the shear of faster-moving fluid particles over slower-moving fluid particles and there is a momentum transfer from the faster to the slower-moving fluid particles. This can be further justified by saying that momentum is transferred in the direction of the velocity gradient. This is further explained in Eqn. 9.

In complicated flows, a model for the large-scale motion of inhomogeneous turbulence which enables the stresses in the turbulence to be estimated, was given in [61].

The standard $k-\epsilon$ model was presented in [62], in which the local turbulent viscosity was calculated from the solution of transport Equations for the turbulence kinetic energy and the rate of energy dissipation. A form of the model was provided that is appropriate for regions with low turbulence Reynolds numbers. The model had been employed to predict wall boundary-layer flows where streamwise accelerations are so significant that the boundary layer partially turns back to laminar and the turbulence structure is directly affected by viscosity. The application of this model in some practical cases like isothermal low Reynolds number pipe flows, wall boundary layers with streamwise pressure gradient and wall injection are detailed in [63]. It has been observed that the structure of the viscous sublayer exhibits a notable influence on the flow of a variety of

turbulent shear flows. The turbulence model used here, calculates the turbulence energy and turbulence dissipation rate using transport Equations that are parallelly solved with the conservation Equations for the mean flow. As a place of improvement, the authors suggested adjustment in the model coefficient, C_μ and similar refinements on the other viscosity dependent terms. The provision of a transport Equation for the turbulent shear stress in the mean momentum Equation was also suggested, hence the variables k and ϵ would remain the unknowns, necessitating the retention of the transport Equations for these variables as well. Such a model was proposed [64], but applicable specifically to regions where the viscosity has no direct impact on turbulence. The discrepancy and the evolution of the standard $k-\epsilon$ model is further addressed in [65] and [66]. The numerical predictions of turbulent flow are reviewed in [65] and showed that turbulence models that calculate the magnitudes of two turbulence quantities, the turbulence kinetic energy k and its dissipation rate ϵ, from transport Equations in tandem solved with those governing the mean flow behavior are best suited for computational economy, range of applicability, and physical realism. The numerical applicability of the model was exemplified by using numerical calculations of nine significantly different types of turbulent flow. This type of model allowed for the prediction of both near-wall and free-shear flow phenomena without requiring modifications to constants or functions and successfully accounts for many low Reynolds number features of turbulence whose application had resulted in reliable estimates of flows with recirculation in addition to those of the boundary-layer type. The wall functions used were based on the assumption that the length scale is a universal function of distance from the wall. The low Re formulation was given here. In [66], the flow generated by a rotating disc in a quiescent atmosphere was considered that produces very high gradients of swirl velocity in the proximity of the disc which in turn highlights previously unnoticed terms in the kinetic energy and dissipation Equations. This application thus provided a test of the model's universality for a significant class of fluid flows. It has been discovered to accurately forecast the flow, heat, and mass transfer in the vicinity of a revolving disc to predict certain low-Reynolds-number phenomena in boundary layers and duct flows, the outcome of which is important for estimating convective heat transfer rates in turbine discs. The CFD software like OpenFOAM, ANSYS Fluent, CFX, Star CCM uses the model coefficients given in [66].

The $k-\omega$ model originated as a development of the $k-\epsilon$ model as the latter cannot accurately predict boundary layers with adverse pressure gradients like that required in aerodynamics and turbomachinery, for instance separation in the trailing edge of an aerofoil for high angle of attack and in diffusing sections where the area increases. If $k-\epsilon$ model is used then the point of separation of flow is predicted incorrectly and thus there are huge differences in the predicted flow parameters like coefficient of lift and drag than the actual ones. To overcome the above-mentioned problem, various other turbulence models were developed with time, the first one by [67]. Many literatures have shown that small changes in freestream turbulent kinetic energy led to large changes in the turbulent viscosity and skin friction coefficient and the absence of cross diffusion

term is the reason for which various values of the coefficients has been suggested by [68]. This will affect the forces on the body and flow separation inception. Now, for an adverse pressure gradient, the point of flow separation occurs where the coefficient of skin friction is zero and this depends on the freestream turbulence conditions. The remedy for this limitation of both the models was proposed [69]. A blend of both the models is used by using the $k-\epsilon$ model far away from the wall in the free stream when it is not susceptible to small changes in k and ω; and then near the wall the $k-\omega$ model is used. In between these, a blend of both the models is used which forms the basis of the $k-\omega$ SST model [70]. Some changes in the two-Equation model Equations, turbulent eddy viscosity, constants and auxiliary functions of the $k-\omega$ model are given in [71]. The one-Equation Spalart-Allmaras model was proposed at this time [72] as an improvement over the k - ε model, to build a model that can predict boundary layer for adverse pressure gradients, which is even complicated when shocks are present, like that in the case of supersonic flow. The $k-\omega$ model is further revisited in [73] in order to address and improve its sensitivity to free-stream boundary conditions on omega. The addition of only one new closure coefficient and a modification to the influence of eddy viscosity on turbulence parameters has been done, due to which, a substantially enhanced model that applies to both boundary layers and free shear flows and has very minimal sensitivity to finite freestream boundary constraints on turbulence parameters has been developed. The model outperforms previous versions in terms of accuracy for even more intricate separated flows.

A new model dissipation rate Equation based on the dynamic Equation for fluctuating vorticity and a new realizable eddy viscosity formulation and a new eddy viscosity formulation had been proposed [74]. These Equations ascertain realizability and hold the effect of mean rotation on turbulence stresses. Applications are in rotating homogeneous shear flows, channel and flat boundary layer flows with and without pressure gradients, boundary-free shear flows and backward facing step flows. This model is studied to be numerically more stable in turbulent flow calculations as the spreading rate anomaly of planar and round jets was removed completely.

Some of the eddy-viscosity models used in this work are: $k-\epsilon$, $k-\omega$, $k-\omega$ SST, realizable $k-\epsilon$, Spalart Allmaras models (RANS) and Smagorinsky, Dynamic Smagorinksy models (LES). A detailed explanation of statistical turbulence modelling is given in [75]. The journey from general Navier-Stokes Equation to the closure of Reynolds stresses and solving of unknown scalar fluxes through various turbulence modelling is given in this book. Another detailed explanation regarding turbulent flows, RANS, LES and Probability Density Function (PDF) are given in [76]. The mathematical concepts and model Equations are developed for various approaches. In this book, it is concluded that the suitability of a particular model depends on a weighted combination of various criteria specific to a particular turbulent flow problem.

There are certain limitations to the Eddy Viscosity models mentioned in [77]. A cubic relation between the strain, vorticity, and stress tensor was proposed in this study which captures the effects of streamline curvature across a variety of flows better than the traditional eddy-viscosity scheme. The variety of flows taken into account range from pipe flow and simple shear at high strain rates to flows with severe streamline curvature and stagnation. The processing time needed for this kind of cubic closure was found to be just 10% longer than for a linear EVM. Other turbulence closure models along with cubic $k-\epsilon$ model is Reynolds stress, Full Sub-grid scale models.

Hence Section 2.1 to 2.4 gives a brief idea of the works on VIV done in the past and the development of various RANS turbulence models over time.

2.5 THE OBJECTIVE AND THE APPROACH

A major design consideration in offshore oil and gas engineering, especially for many floating offshore structures to collect oil and gas from the deep sea, is vortex-induced vibration (VIV) of cylinders. This is the main objective of the thesis work. The approach to study this significant issue is explained next and can be sub-divided into major 3 parts.

Part-1: It is worth noting that understanding the underlying mathematics, involved algorithms and flow physics thoroughly is crucial. This is why, solving of some canonical flow problems using in-house MATLAB coding is necessary initially to call for a clear understanding of both fluid dynamics and programming. It is imperative to unambiguously *specify the category* of canonical flow issue that is required to be solved. The categories typically comprise potential flow, viscous flow, incompressible flow, and compressible flow. The task at hand involves identifying the *boundary conditions*, fluid properties, and any simplifications or assumptions that can be made to facilitate the management of the problem. *Developing a mathematical model* involves formulating governing equations that accurately describe the fluid flow behavior in accordance with the defined problem. In the context of the present work, the Navier-Stokes equations are utilized to tackle viscous flows. *Discretization* involves the partitioning of the computational domain into a grid or mesh, which enables the numerical solution of equations. The prevalent techniques employed in this context encompass finite difference, finite element, and finite volume methods. A suitable discretization method that aligns with the problem's attributes and computational demands must be chosen. In order to effectively address the problem at hand, it is necessary to implement appropriate *boundary conditions*. These conditions include the specification of velocity or pressure at specific boundaries. The system's boundary conditions can be incorporated by adjusting the equations or implementing appropriate numerical techniques. *The iterative methodology* is to be employed to solve the equations by setting up a loop that iterates through the discretized domain. The approach involves initiating an initial estimation for the solution, followed by the application of numerical techniques such as Gauss-Seidel or successive over-relaxation to iteratively update the solution. Iterations are carried forward until the convergence criteria have been satisfied. Upon achieving convergence of the solution, it is necessary to extract and visualize the pertinent outcomes through post-processing. Performing computations of various parameters, including but

not limited to velocity profiles, pressure distributions, and streamline patterns follows MATLAB's plotting functions to graphically represent the outcomes in two or three dimensions, and produce any essential diagrams or illustrations. To ensure the precision of the developed code, it is essential to *compare the findings* with analytical solutions, experimental data, or established benchmark cases. This process of validation and refinement helps to verify the accuracy of the code. In the event that inconsistencies are detected, the plausible origins of inaccuracies are found to subsequently enhance the execution of the task. In the realm of computer programming, *codes are optimized* for efficiency, particularly when handling larger or more intricate problems, in order to enhance performance. MATLAB provides a range of methodologies such as vectorization, parallel computing, and leveraging the built-in linear algebra functions to enhance computational efficiency. In order to ensure proper documentation and reusability of code, the codes are thoroughly documented by including comments that explain the purpose of each section and its underlying equations. The present work is documented with the aim to facilitate comprehension and future reuse or sharing of the code.

Part-2: To extend the work mentioned in Part-1 in a much more robust manner, an open-source C++ based solver, namely OpenFOAM is used to simulate the fluid flow. With the help of various solver types, dictionaries, mathematical schemes, the vortex-induced vibration of an elastically mounted cylinder is captured.

In this study, at first one-degree-of-freedom (1DOF) VIV of an elastically mounted rigid circular cylinder of a finite length, allowed to vibrate in the translational or cross-flow direction (here, y-direction) is studied numerically. Then the same is studied for two-degrees-of-freedom (2DOF) at the reduced velocity where the amplitude is maximum for 1DOF system. The 2DOF system considered is allowed to oscillate in the inline (x-direction) and cross-flow directions (y-directions). This setup is in some extent similar to a simplified configuration of an offshore spar platform used in the offshore oil and gas engineering. The amplitude (response) of the cylinder, frequency, along with the force coefficients with respect to a range of reduced velocity, V_r are computed in this study. The reduced velocity is a non-dimensional velocity, given by the ratio of free-stream velocity (U) to the product of natural frequency of the structure (f_n) and characteristic diameter of the cylinder (here, a circular cylinder, D).

The 1DOF and 2DOF movement of the cylinder is allowed by the installation of notional springs with particular stiffness. This stiffness varies according to various reduced velocities. But the stiffness is dependent on its value, arrangement (series or parallel) and the location of the springs when computed numerically. This study is not explicitly done in any paper. The variation of amplitude of the cylinder due to the location, arrangement and value of the stiffness of springs, thus, raising the question whether the unrestrainment of the cylinder in the form of putting springs is absolutely notional is studied first for the case of 1DOF. Also, the effect of the free-end of the cylinder on the response of the cylinder is studied here. Thus, this work focuses on two types of studies:

i) For cylinders of low-aspect ratio, the effects of the free-end of 1DOF cylinder that leads to the vortex shedding convection towards the top end of the cylinder by the upwash velocity, as mentioned in [20] is also studied in this work.

ii) The same have been studied for 2DOF VIV of cylinder at the reduced velocity where the amplitude is maximum for Case-(i).

Similar to [20], the 'downwash' velocity is named as 'upwash' velocity in this study, because the free-end of the cylinder faces downward and the top surface of the cylinder is in the same level of the surface of water. Only f_n, or equivalently, the stiffness k of the spring restraining the cylinder, is altered to keep Reynolds number constant across cases while V_r varies.

Thus, the present work is expected to contribute to a better understanding of the responses of a FOWT (FOWT is Floating Offshore Wind Turbine) to vortex shedding in current, also known as vortex-induced motion (VIM) in the field of offshore engineering, and will provide direction for the design of FOWTs.

Part-3: Till this, a maximum of Reynolds number 300 has been used, as it is already in the turbulent regime for the case of a fluid flow-induced oscillating bluff body. So, for a static square cylinder, large eddy simulations (LES) have been performed next to find the drag and lift at Reynolds number 10^5.

To achieve this quite a challenging problem, the present work has been divided into certain parts, as detailed in Chapter-5.

CHAPTER 3: IN-HOUSE CODING AND MATHEMATICAL DESCRIPTION

3.1 INTRODUCTION

This chapter delves into the mathematical formulation of the current workflow. Initially, the non-linear Navier Stokes equations are computed for several canonical flow scenarios, including a) fluid motion within a lid-propelled cavity, b) fluid motion over a stationary square-shaped cylinder, and c) vibration caused by vortices on a square-shaped cylinder. The initial subsection provides the respective mathematical formulations. These flow problems are solved using in-house MATLAB codes that are developed within a finite difference (FD) framework. Next, the problem of vortex-induced vibration (VIV) in a cylindrical offshore spar platform is addressed with the finite volume (FV) method in OpenFOAM, which provides the necessary level of robustness. In this instance, the utilization of the open-source flow solver OpenFOAM, which is based on the C++ programming language, is employed. The Reynolds-averaged Navier Stokes (RANS) equations are utilized to tackle turbulence in the flow domain. The subsequent subsection outlines the corresponding mathematical formulations. The subsequent section expounds upon the Large Eddy Simulation (LES) methodology. The high Reynolds number flow-past square cylinder problem is addressed through the utilization of the Large Eddy Simulation (LES) technique within the OpenFOAM computational framework.

3.2 CANONICAL FLOW PROBLEMS THROUGH IN-HOUSE MATLAB CODING

The utilization of Computational Fluid Dynamics (CFD) is a potent numerical methodology employed for the examination and evaluation of canonical fluid flow phenomena. This approach offers a mechanism for emulating and forecasting fluid dynamics by resolving the governing equations of fluid mechanics through numerical techniques. The lid-driven cavity and flow past cylinder problems are frequently employed as standard flow problems to showcase the capabilities of computational fluid dynamics (CFD). In the present work, the canonical flow problems are solved using in-house MATLAB coding.

3.2.1 Lid-Driven Cavity

The Lid-Driven Cavity Problem pertains to a fluid-filled square cavity, wherein the top lid is subjected to a predetermined velocity while the remaining walls remain motionless. This issue embodies a fundamental instance in the field of fluid mechanics, showcasing the generation of vortices and the flow properties within a confined area.

The utilization of computational fluid dynamics (CFD) involves the application of the Navier-Stokes equations, which are fundamental in describing the movement of viscous fluids, to address the lid-driven cavity problem. The equations can be expressed as follows for flows that are incompressible:

- Continuity equation:

$$\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} = 0 \qquad [3.1]$$

- x-Momentum equation:

$$\frac{\partial u}{\partial t} + \frac{\partial uu}{\partial x} + \frac{\partial vu}{\partial y} + \frac{\partial wu}{\partial z} = -\frac{1}{\rho}\frac{\partial p}{\partial x} + \frac{\mu}{\rho}\nabla^2 \mathrm{u} \qquad [3.2]$$

- y-Momentum equation:

$$\frac{\partial v}{\partial t} + \frac{\partial uv}{\partial x} + \frac{\partial vv}{\partial y} + \frac{\partial vw}{\partial z} = -\frac{1}{\rho}\frac{\partial p}{\partial y} + \frac{\mu}{\rho}\nabla^2 \mathrm{v} \qquad [3.3]$$

- z-Momentum equation:

$$\frac{\partial w}{\partial t} + \frac{\partial uw}{\partial x} + \frac{\partial vw}{\partial y} + \frac{\partial ww}{\partial z} = -\frac{1}{\rho}\frac{\partial p}{\partial z} + \frac{\mu}{\rho}\nabla^2 \mathrm{w} \qquad [3.4]$$

The symbols u, v, w denote the x, y, z components of the velocity vector, p represents the pressure, ρ signifies the fluid density, μ stands for the dynamic viscosity, and the symbol ∇ denotes the gradient operator.

The boundary conditions for the lid-driven cavity problem encompass the no-slip condition on the walls ($u = 0$), where the velocity is zero, and the prescribed velocity condition on the moving lid, where the velocity is U, a specified value.

The utilization of computational fluid dynamics (CFD) involves the discretization of the computational domain into a grid or mesh, followed by the iterative solution of the Navier-Stokes equations over the mesh until convergence is attained. The aforementioned procedure entails the modification of the velocity and pressure fields through the utilization of discretized equations, while also taking into account the boundary conditions to replicate the flow characteristics within the cavity.

3.2.2 Flow Past a Static Square Cylinder

The problem of flow past a cylinder involves the simulation of fluid flow around a circular cylinder that is positioned within a uniform flow. This phenomenon is frequently employed in the

investigation of fluid flow separation, the emergence of vortices (such as the Von Kármán vortex street), and the drag and lift forces experienced by the cylinder.

The Navier-Stokes equations are revisited for the case of incompressible flow around a cylinder. The continuity equation can be expressed as the divergence of the velocity field being equal to zero (Eq. 3.1). The governing equations of momentum, can be expressed as Eq. 3.2 to Eq. 3.4. Boundary conditions are a crucial aspect of the flow past cylinder problem. Typically, these conditions involve the specification of incoming flow conditions, such as uniform velocity, and the application of the no-slip condition ($u = 0$) on the surface of the cylinder.

Computational fluid dynamics (CFD) methodologies, including the finite difference approach, are employed to discretize the computational domain encompassing the cylinder into a mesh. Iterative methods are employed to solve the Navier-Stokes equations, which incorporate the convective term, over the mesh in order to derive the flow field. This facilitates the examination of flow patterns, pressure distribution, drag and lift forces, and other pertinent parameters.

3.2.3 Vortex-Induced Vibration (VIV)

The mathematical representation of Vortex-Induced Vibration (VIV) encompasses two fundamental components, namely, fluid flow simulation and structural dynamics analysis. The subsequent passage presents a depiction of the mathematical formulation in MATLAB from a detached viewpoint.

Simulation of Fluid Flow

The process involves the definition and discretization of the computational domain into a mesh comprising of control volumes. The fluid characteristics, namely density (ρ) and dynamic viscosity (μ), have been designated. The initial and boundary conditions have been established, encompassing the inlet flow conditions and the solid boundaries. The Navier-Stokes equations are commonly solved through the utilization of suitable numerical techniques, such as finite difference or finite volume methods, which enable the acquisition of the fluid velocity and pressure fields.

The computation of the fluid forces exerted on the structure is predicated upon the pressure distribution derived from the flow simulation.

Structural Dynamic Analysis

The fundamental characteristics of a system's structure, such as its mass (m), stiffness (k), and damping (c), are explicitly specified. The structural motion is initiated by setting initial conditions, including displacement and velocity. The forces that are derived from the simulation of fluid flow are subsequently applied onto the structure. Numerical integration techniques, such as the

Newmark-β or Crank-Nikolson methods, are employed to solve the equation of motion for the structure, resulting in the determination of the structural displacement and velocity responses. The fluid flow simulation undergoes updates in accordance with the latest changes in the structural position and velocity. The iterative process of fluid-structure interaction is executed until the point of convergence or the designated simulation duration is attained.

The Semi-Implicit Method for Pressure-Linked Equations (SIMPLE)

The Semi-Implicit Method for Pressure-Linked Equations (SIMPLE) is a prevalent numerical technique in the field of Computational Fluid Dynamics (CFD) that is utilized for resolving incompressible flow predicaments. The methodology offers a methodical framework for resolving the Navier-Stokes equations. The SIMPLE algorithm is a widely used computational method in fluid dynamics. It stands for Semi-Implicit Method for Pressure-Linked Equations and is used to solve the Navier-Stokes equations, which describe the motion of fluids. The algorithm involves a series of iterative steps that update the velocity and pressure fields until a converged solution is obtained. It is known for its ability to handle complex geometries and turbulent flows. The SIMPLE algorithm has been implemented in various software packages and is commonly used in engineering applications.

The initial phase of the algorithm involves defining the computational domain and discretizing it into a grid or mesh. The velocity components (u, v, w) and pressure (p) are assigned initial values across the entire domain, and boundary conditions for velocity and pressure are specified.

In the prediction phase, the algorithm utilizes interpolation from the cell center values to estimate the velocities at the cell faces. The convective terms in the momentum equations are computed by utilizing the interpolated velocities. Subsequently, the equations governing momentum are solved to derive the anticipated velocities at the succeeding time increment.

Subsequently, the pressure correction phase ensues. The introduction of a pressure correction is necessary due to the potential discrepancy between the predicted velocities and the continuity equation. A pressure correction equation has been derived by utilizing the continuity equation and integrating the divergence of the anticipated velocities. The determination of the correction values for pressure can be achieved by solving the aforementioned equation.

Subsequent to acquiring the pressure correction, the algorithm proceeds to revise the velocities and pressure. The anticipated velocities are modified through the deduction of the pressure correction, thereby guaranteeing their compliance with the continuity equation. The pressure is modified through the addition of a correction term to the preceding pressure measurements.

During each iteration, the algorithm verifies whether convergence has been achieved. Commonly, this process entails the assessment of the residual values associated with the continuity equation,

momentum equations, or pressure correction equation. In the event of non-convergence, the algorithm iterates through the prediction, pressure correction, and update steps until the convergence criteria are satisfied.

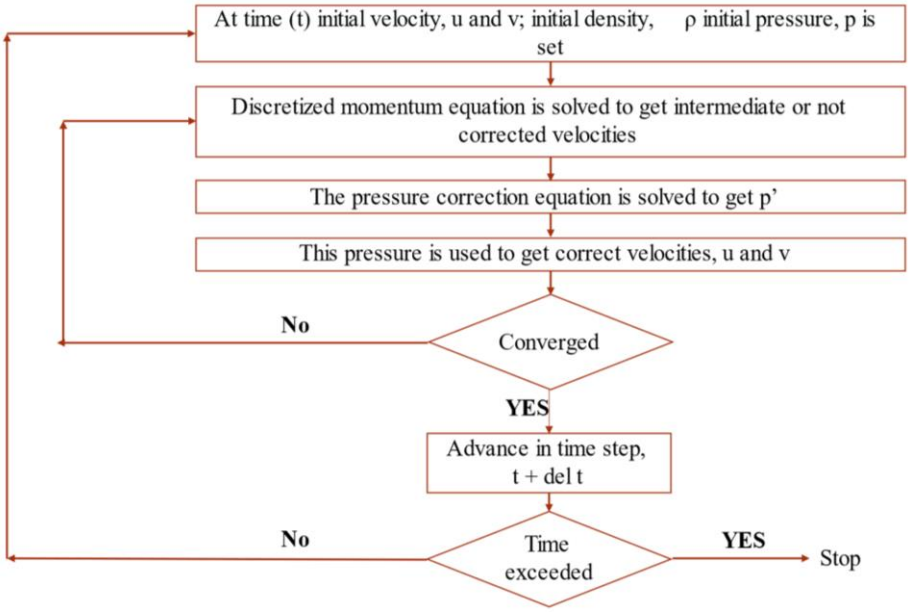


Fig. 3.1: Solution methodology of Navier-Stokes equation using Semi-Implicit Method for Pressure-Linked Equations (SIMPLE) algorithm

The SIMPLE algorithm is a dependable and effective methodology for addressing incompressible flow issues through the iterative coupling of velocity and pressure fields. The methodical procedure employed guarantees the fulfillment of the continuity equation and facilitates precise modelling of fluid dynamics. Several adaptations of the SIMPLE algorithm, namely SIMPLER, and PISO, have been devised to tackle distinct flow scenarios and augment stability and convergence.

3.3 VORTEX-INDUCED VIBRATION (VIV) OF 3D OFFSHORE SPAR PLATFORM

The occurrence of self-excited oscillations in a fluid-immersed structure due to the interaction between the fluid flow and the structure is known as vortex-induced vibration (VIV). The phenomenon of vortex-induced vibration (VIV) holds noteworthy consequences for the development and evaluation of engineering constructions, including but not limited to offshore platforms, risers, and marine cables. The utilization of Computational Fluid Dynamics (CFD) in conjunction with OpenFOAM, a CFD software package that is open-source, presents a robust mechanism for the simulation and examination of Vortex-Induced Vibration (VIV) issues.

In order to mathematically express the vortex-induced vibration (VIV) problem utilizing OpenFOAM, the governing equations are derived from the Navier-Stokes equations that describe fluid motion, and are coupled with the equations of structural dynamics. The subsequent procedures delineate the mathematical formulation.

Fluid Flow Simulation

The process of fluid flow simulation involves the discretization of the Navier-Stokes equations, which govern the behavior of fluids, through the application of the finite volume method. The discretization of the fluid domain involves partitioning it into a mesh structure, where each individual cell corresponds to a relatively small control volume. Turbulence models, such as the Reynolds-averaged Navier-Stokes (RANS) equations or large eddy simulation (LES), are utilized to effectively capture the characteristics of turbulent flow. The specified boundary conditions encompass the inlet flow conditions, wall conditions, and outlet conditions.

Structural Dynamics

The dynamics of a structure are determined by equations of structural dynamics, such as the equation of motion for a rigid or flexible body. The discretization of equations is accomplished through the utilization of suitable techniques, such as the finite element method (FEM) in the case of flexible structures. The non-linear finite difference bases motion-solving techniques such as Newmark-β, Crank-Nikolson, or Symplectic are employed in the case of the rigid body. The structure is assigned material properties, including stiffness and damping coefficients. Boundary conditions are established by specifying fixed points or prescribed displacements at particular locations.

Fluid-Structure Interaction

Fluid-Structure Interaction (FSI) is a computational technique that involves coupling the fluid and structural domains through FSI algorithms. The pressure distributions derived from the fluid flow simulation are utilized to compute the forces imposed by the fluid on the structure.

The structure is subjected to external forces that impact its motion, thereby causing deformation which in turn has an impact on the fluid flow field. The FSI iterative process is executed until convergence is attained, wherein the fluid and structure domains exchange information during each iteration. Upon achieving convergence of the simulation, it is possible to extract diverse quantities of interest from the obtained outcomes. In the context of Vortex-Induced Vibration (VIV) problems, pertinent variables encompass the structural displacement, velocity, and forces, alongside the vortex-shedding configuration and flow properties.

The equation of motion pertains to the study of the rigid body's dynamics in reaction to the fluid forces that are exerted on it. The equation of motion for a rigid body experiencing vortex-induced vibration (VIV) can be represented as

$$m\left(\frac{d^2r}{dt^2}\right) = F_v + F_d \quad [3.5]$$

where m denotes the mass of the body, r represents the displacement of the body, and F_v and F_d denote the forces of vortex shedding and damping, respectively.

The phenomenon known as Vortex-Induced Force (VIF) refers to the force that is generated when a fluid flow interacts with a bluff body, such as a cylinder or a sphere, resulting in the formation of vortices in the wake of the body. The phenomenon of vortex-induced force is a result of the interplay between the fluid dynamics and the solid structure. The given entity can be resolved into two distinct constituents, namely the force responsible for lifting F_{lift} and the force that opposes the motion of the object through a fluid medium F_{drag}. The force induced by a vortex can be expressed as the sum of the lift force and the drag force.

The lift force, which is produced by the periodic shedding of vortices around the object, is oriented perpendicular to the direction of fluid flow. The formula for lift force can be expressed as follows:

$$F_{lift} = 0.5\,\rho A C_l V^2 \quad [3.6]$$

where ρ represents air density, A represents the surface area of the object, C_l represents the lift coefficient, and V represents the velocity of the object.

The drag force is a vector quantity that is oriented in the same direction as the fluid flow and acts in opposition to the movement of the object. The formula for drag force can be mathematically represented as follows:

$$F_{drag} = 0.5\,\rho A C_d V^2 \quad [3.7]$$

where ρ denotes the density of the fluid, A represents the cross-sectional area of the object, Cd is the drag coefficient, and V is the velocity of the object.

The drag coefficient, denoted by C_d, is contingent upon both the flow conditions and the geometry of the body.

The aforementioned equations present a mathematical representation of the vortex-induced vibration (VIV) phenomenon in the context of a solid object.

OpenFOAM solves the fluid flow, rigid body motion, and the coupled fluid-structure interactions in a finite volume framework. Therefore, it will be relevant to discuss this technique.

3.3.1 Finite Volume Method (FVM)

The finite volume method is a numerical technique used to solve partial differential equations (PDEs) by discretizing the computational domain into small control volumes or cells. Each cell represents a small portion of the domain where the governing equations are approximated. Here is a brief mathematical description of the finite volume method. One can find more detailing on the same in Annexure-I: Section-3.3.1.

Discretization of the Domain

The computational domain Ω is divided into a set of non-overlapping control volumes or cells, denoted as Ω_i. The boundaries of each cell, denoted as $\partial\Omega_i$, consist of faces or surfaces. The mesh or grid represents the collection of cells and their interconnected faces.

Conservation Laws

Consider a scalar quantity u defined over the domain Ω, which satisfies a conservation law, such as mass, momentum, or energy. The governing PDE for u can be written in integral form as:

$$\int\int \partial\Omega\, F(u) \cdot dA \;=\; \int\int \Omega S(u)\, dV \qquad [3.8]$$

where $F(u)$ represents the flux of u across the cell face, $S(u)$ represents the source term, dA denotes the differential area element, and dV denotes the differential volume element.

Discretization of the Governing Equation

The integral equation is approximated by integrating over each cell Ω_i, yielding:

$$\int\int \partial\Omega_i\, F(u) \cdot dA \;\approx\; \sum\sum \partial\Omega_{ij}\, F(u_{ij}) \;\cdot\; A_{ij} \qquad [3.8]$$

where $\partial\Omega_{ij}$ represents the face shared by cells Ω_i and Ω_j, $F(u_{ij})$ represents the flux evaluated at the face, and A_{ij} represents the area of the face.

Application of the Divergence Theorem

The divergence theorem is applied to transform the surface integral into a volume integral, resulting in:

$$\sum\sum \partial\Omega_{ij}\, F(u_{ij}) \cdot A_{ij} \;=\; \sum\sum \Omega_i\, \nabla \cdot F(u_i)\, V_i \qquad [3.9]$$

where $\nabla \cdot F(u_i)$ denotes the divergence of the flux at cell Ω_i, and V_i represents the volume of the cell.

Approximation of Variables

The variables u and its flux $F(u)$ are approximated within each cell using interpolation or other suitable approximation techniques. The cell-averaged values, denoted as u_i and F_i, are typically used to represent the variables within cell Ω_i.

Discretized Conservation Equation

Substituting the approximated variables into the discretized equation, we obtain the discrete conservation equation for cell Ω_i:

$$\nabla \cdot F_i \approx \frac{1}{V_i} \sum\sum \partial\Omega_{ij} F_{ij} \cdot A_{ij} \qquad [3.10]$$

where F_{ij} represents the interpolated flux value at the shared face between cells Ω_i and Ω_j.

Closure Relations

Closure relations, such as gradient approximations or turbulence models, are used to evaluate the terms $\nabla \cdot F_i$ and F_{ij}.

Assembly of Linear System

The discretized conservation equations for all cells are combined to form a system of linear equations. Additional equations, such as boundary conditions and source terms, are incorporated into the system. The resulting system can be solved using numerical techniques like matrix inversion, iterative solvers, or direct methods.

Iterative Solution and Convergence:

The linear system is solved iteratively until a desired level of convergence is achieved. Convergence is typically assessed based on criteria such as residual error.

The detailed mathematical formulation that is used by OpenFOAM to calculate cell volume using various gradient, time derivative schemes are detailed in Annexure-I: Section-3.3. A finite volume method has been used to discretise the governing equations in Sections-5.3 and 5.4.

A transient, first-order implicit, bounded scheme is used for temporal discretisation while an unbounded Gaussian second-order upwind scheme is adopted for convective terms. Cell-based linear interpolation scheme is used to interpolate cell centre values to face centres. Gradient terms

are computed by a second-order, cell limited Gauss linear scheme. These schemes are explained in Annexure-I: Section-4.3.3 (*fvSchemes*).

3.4 TURBULENCE MODELLING

Reynolds-Averaged Navier-Stokes (RANS) and Large Eddy Simulation (LES) are computational fluid dynamics (CFD) techniques used to simulate turbulent flows. Here is a brief mathematical description of RANS and LES. Annexure-I: Section-3.4 deals with more detailed formulations.

3.4.1 Reynolds-averaged Navier-Stokes (RANS)

RANS is a time-averaged approach that models turbulent flows using statistical averaging. The RANS equations are derived by applying Reynolds averaging to the Navier-Stokes equations. The governing equations for RANS can be written as:

Continuity Equation:

$$\nabla \cdot (\rho \langle u \rangle) = 0 \qquad [3.11]$$

Momentum Equations:

$$\nabla \cdot (\rho \langle u \rangle u) = -\nabla P + \nabla \cdot (\mu(\nabla u + (\nabla u)\hat{}T) - \rho \langle u'u' \rangle) \qquad [3.12]$$

where ρ is the fluid density, $\langle u \rangle$ represents the time-averaged velocity, P is the pressure, μ is the dynamic viscosity, and $\langle u'u' \rangle$ denotes the Reynolds stress tensor. The Reynolds stress tensor is usually modeled using turbulence models, such as the eddy viscosity models (e.g., $k-\varepsilon, k-\omega$), which introduce additional closure relations. RANS simulations are computationally efficient but provide time-averaged information and do not resolve the complete range of turbulent scales.

3.4.2 Large Eddy Simulation (LES)

LES is a time-resolved approach that aims to capture large-scale turbulent structures while modeling the effect of smaller scales. In LES, the flow field is decomposed into resolved large-scale eddies and unresolved small-scale eddies. The governing equations for LES can be written as:
Continuity Equation:

$$\nabla \cdot u = 0 \qquad [3.13]$$

Momentum Equations:

$$\partial u/\partial t + \nabla \cdot (uu) = -\nabla P + \nabla \cdot (\mu(\nabla u + (\nabla u)^T \hat{} T)) - \nabla \cdot \tau \qquad [3.14]$$

where u represents the velocity, P is the pressure, μ is the dynamic viscosity, and τ denotes the subgrid-scale (SGS) stress tensor. The SGS stress tensor τ is typically modeled using SGS models, such as the Smagorinsky model or dynamic models. LES captures a wide range of turbulent scales but requires resolving the smallest scales, making it computationally more expensive than RANS. LES is suitable for simulating flows with significant turbulent structures and unsteady phenomena.

Closure Models

Both RANS and LES require closure models to represent the unresolved turbulent quantities (Reynolds stress in RANS and SGS stress in LES).

Closure models introduce additional equations or assumptions to close the system of equations.

Various closure models exist, including algebraic models, transport equation models, and dynamic models, which have different levels of accuracy and complexity.

Numerical Solution

Both RANS and LES equations are discretized using numerical methods, such as finite volume, finite difference, or finite element methods. Time integration schemes, such as explicit or implicit methods, are used to advance the solution in time. The resulting discretized equations form a system of equations that can be solved using iterative or direct methods.

CHAPTER 4: NUMERICAL METHODOLOGY

4.1 INTRODUCTION

Numerical procedures are essential for engineering and scientific research, as they allow researchers to examine and understand complicated processes. MATLAB is a well-liked tool in the scientific and engineering fields due to its adaptability and a broad selection of integrated functions and toolboxes. Researchers can create numerical models and algorithms using MATLAB to solve canonical flow problems, such as lid-driven cavity, flow past cylinders, etc. OpenFOAM is a prevalent computational fluid dynamics (CFD) application used for modelling and deriving insights from vortex-induced vibration (VIV). The finite volume method describes flow behaviour and the interaction with solid structures. Post-processing features allow for viewing and analysis of the simulation results, assisting researchers in determining crucial parameters and enhancing system performance.

Finite difference and finite volume methods have been used in the works presented in this thesis. For the canonical flow problems, finite difference method has been used. A staggered grid configuration is used to simulate the flow within lid-propelled cavity. The main idea of staggered grid configuration, shown in Fig. 4.2.1, is that every cell is either square or rectangular and contains a particular address, which facilitates the usage of finite difference with various differencing schemes, for this type of grid system. It uses the Cartesian mesh properties. The cell parameters (vectors) are stored in the cell faces instead of the cell centres. This is advantageous with respect to the collocated grid system that stores the cell parameters at the cell centres because it does not give rise to chequerboard oscillations. Thus, staggered grid configuration gives more stable solutions.

But the flow problems solved in OpenFOAM as detailed in Section 4.2 and 4.3, a collocated grid system is used. This is because there are quite a number of disadvantages of staggered grid configuration. Due to its usage of rectangular or square cells only, it cannot suffice the meshing of any geometry with continuous smooth edges. Moreover, staggered grid contains some ghost cells that lead to larger errors than chequerboard oscillations, when two edges of the geometry are at a particular angle. Thus, staggered grid configuration is useful only for 2D cases. These are the reasons why most of the CFD software or solvers use collocated grid configuration.

In Section 4.3, the most important part of vortex-induced vibration of cylinders is the dynamicMeshDict where the structure properties, constraints and restraints are specified. This dictionary distinguishes the case from that of flow past a static cylinder. Finite volume discretization is used in the case of Section 4.3 and 4.4.

4.2 CANONICAL FLOW PROBLEMS THROUGH IN-HOUSE MATLAB CODING

Finite difference method has been used to solve the 2D Navier-Stokes equations using in-house MATLAB code. The problem statements are detailed in Section 5.2 of Chapter-5. The staggered grid configuration used while solving the Navier-Stokes equation is shown in Figure-4.2.1. 'u', 'v' and 'p' represent the horizontal component of velocity, vertical component of velocity and pressure at each cell. The cell addresses are denoted at the subscripts of these notations.

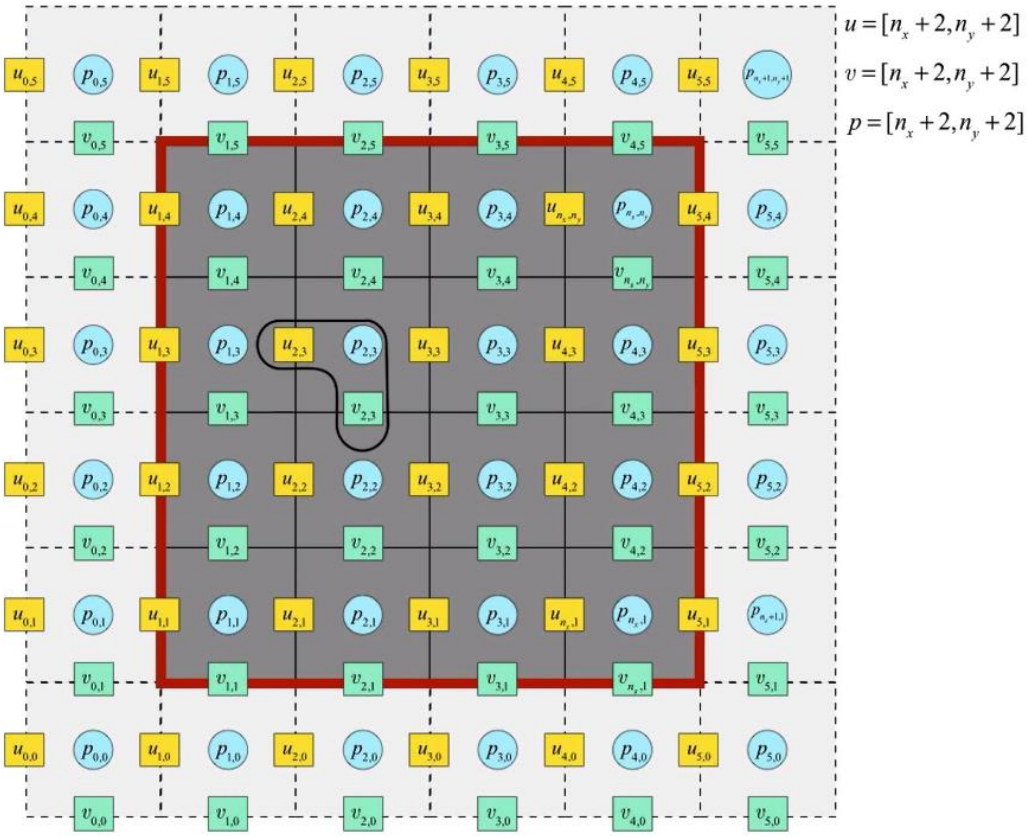


Fig. 4.2.1: Staggered grid system (source: http://www.tonysaad.net/ucfd/)

The finite difference scheme used to discretize the Navier-Stokes equations are shown next.

Discretization of X-Momentum Equations Considering Staggered Grid System

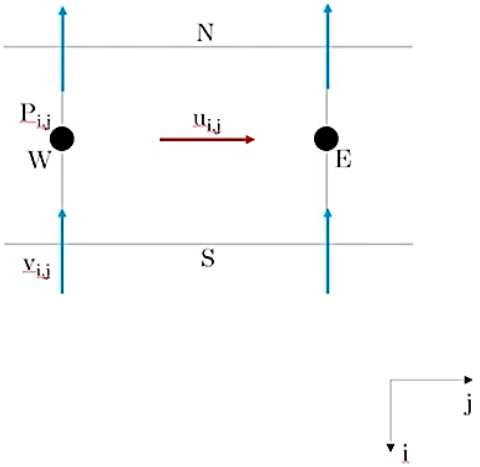


Fig. 4.2.2: Discretization of X-Momentum

$$\iint u\frac{\partial u}{\partial x}dx\,dy = \left[u_E\left(\frac{u_{i,j}+u_{i,j+1}}{2}\right) - u_W\left(\frac{u_{i,j}+u_{i,j-1}}{2}\right)\right]\Delta y \quad [4.1]$$

$$\iint v\frac{\partial u}{\partial y}dx\,dy = \left[v_N\left(\frac{u_{i,j}+u_{i-1,j}}{2}\right) - v_S\left(\frac{u_{i,j}+u_{i+1,j}}{2}\right)\right]\Delta x \quad [4.2]$$

$$\frac{1}{Re}\iint \frac{\partial^2 u}{\partial x^2}dx\,dy = \frac{1}{Re}\left[\left(\frac{u_{i,j+1}-u_{i,j}}{\Delta x}\right) - \left(\frac{u_{i,j}-u_{i,j-1}}{2}\right)\right]\Delta y \quad [4.3]$$

$$\frac{1}{Re}\iint \frac{\partial^2 u}{\partial y^2}dx\,dy = \frac{1}{Re}\left[\left(\frac{u_{i-,j}-u_{i,j}}{2}\right) - \left(\frac{u_{i,j}-u_{i+1,j}}{2}\right)\right]\Delta x \quad [4.4]$$

$$\iint -\frac{\partial p}{\partial x}dx\,dy = -\Delta y(p_{i,j+1} - p_{i,j}) \quad [4.5]$$

$$a_e u_{i,j} = a_E u_{i,j+1} + a_W u_{i,j-1} + a_N u_{i-1,j} + a_S u_{i+1,j} + d_e(p_{i,j+1} - p_{i,j}) \quad [4.6]$$

$$a_e = \frac{u_E}{2}\Delta y - \frac{u_W}{2}\Delta y + \frac{v_N}{2}\Delta x - \frac{v_S}{2}\Delta x + \frac{1}{Re} + \frac{1}{Re} + \frac{1}{Re} + \frac{1}{Re};\ d_e = -\frac{\Delta y}{a_e} \quad [4.7]$$

$$a_E = -\frac{u_E}{2}\Delta y + \frac{1}{Re};\ a_W = \frac{u_W}{2}\Delta y + \frac{1}{Re};\ a_N = -\frac{v_N}{2}\Delta x + \frac{1}{Re};\ a_S = \frac{v_S}{2}\Delta x + \frac{1}{Re} \quad [4.8]$$

Discretization Of Y-Momentum Equations Considering Staggered Grid System

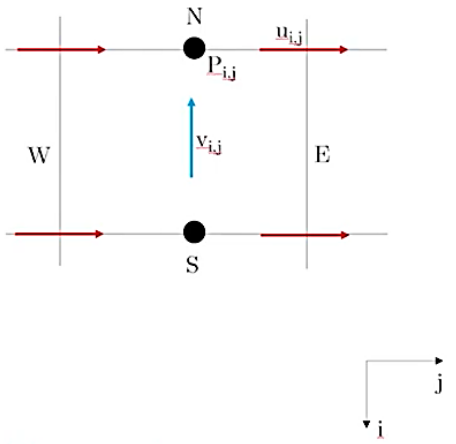


Fig. 4.2.3: Discretization of Y-Momentum

$$\iint u\frac{\partial v}{\partial x}dx\,dy = \left[u_E\left(\frac{v_{i,j}+v_{i,j+1}}{2}\right) - u_W\left(\frac{v_{i,j}+v_{i,j-1}}{2}\right)\right]\Delta y \quad [4.9]$$

$$\iint v\frac{\partial v}{\partial y}dx\,dy = \left[v_N\left(\frac{v_{i,j}+v_{i-1,j}}{2}\right) - v_S\left(\frac{v_{i,j}+v_{i+,j}}{2}\right)\right]\Delta x \quad [4.10]$$

$$\frac{1}{Re}\iint \frac{\partial^2 v}{\partial x^2}dx\,dy = \frac{1}{Re}\left[\left(\frac{u_{i,j+1}-u_{i,j}}{\Delta x}\right) - \left(\frac{u_{i,j}-u_{i,j-1}}{\Delta x}\right)\right]\Delta y \quad [4.11]$$

$$\frac{1}{Re}\iint \frac{\partial^2 v}{\partial y^2}dx\,dy = \frac{1}{Re}\left[\left(\frac{u_{i-,j}-u_{i,j}}{\Delta y}\right) - \left(\frac{u_{i,j}-u_{i+1,j}}{\Delta y}\right)\right]\Delta x \quad [4.12]$$

$$\iint -\frac{\partial p}{\partial y}dx\,dy = -\Delta(p_{i,j} - p_{i+1,j}) \quad [4.13]$$

$$a_n v_{i,j} = a_E v_{i,j+1} + a_W v_{i,j-1} + a_N v_{i-1,j} + a_S v_{i+1,j} + d_n(p_{i-1,j} - p_{i,j}) \quad [4.14]$$

$$a_n = \frac{u_E}{2}\Delta x - \frac{u_W}{2}\Delta x + \frac{v_N}{2}\Delta y - \frac{v_S}{2}\Delta y + \frac{1}{Re} + \frac{1}{Re} + \frac{1}{Re} + \frac{1}{Re};\ d_n = -\frac{\Delta x}{a_n} \quad [4.15]$$

$$a_E = -\frac{u_E}{2}\Delta + \frac{1}{Re};\ a_W = \frac{u_W}{2}\Delta x + \frac{1}{Re};\ a_N = -\frac{v_N}{2}\Delta y + \frac{1}{Re};\ a_S = \frac{v_S}{2}\Delta y + \frac{1}{Re} \quad [4.16]$$

Pressure Correction Equations Considering Staggered Grid System

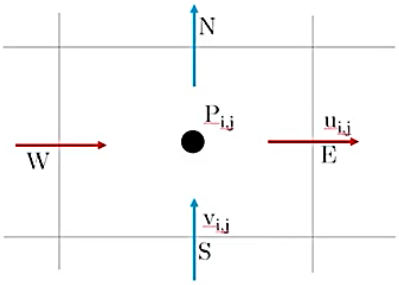


Fig. 4.2.4: Discretization of Y-Momentum

$$\iint \frac{\partial u}{\partial x} dx\, dy = (u_{i,j} - u_{i,j-1})\Delta y \quad [4.17]$$

$$\iint \frac{\partial v}{\partial x} dx\, dy = (v_{i,j} + v_{i-1,j})\Delta y \quad [4.18]$$

$$u_{i,j} = u^*_{i,j} + d_n(p'_{i,j+1} - p'_{i,j}) \quad [4.19]$$

$$u_{i,j-1} = u^*_{i,j-1} + d_w(p'_{i,j} - p'_{i,j-1}) \quad [4.20]$$

$$v_{i,j} = v^*_{i,j} + d_n(p'_{i-1,j} - p'_{i,j}) \quad [4.21]$$

$$v_{i,j} = v^*_{i,j} + d_s(p'_{i,j} - p'_{i+1,j}) \quad [4.22]$$

$$a_p p'_{i,j} = a_E p'_{i,j+1} + a_W p'_{i,j-1} + a_N p'_{i-1,j} + a_S p'_{i+1,j} + b \quad [4.23]$$

$$a_p = -d_e\Delta y - d_w\Delta y - d_n\Delta x - d_s\Delta x \quad [4.24]$$

$$b = -\left(u^*_{i,j} - u^*_i, j-1\right)\Delta y + \left(v^*_{i,j} - v^*_{i,j-1}\right)\Delta x \quad [4.25]$$

$$a_E = -d_e\Delta y;\ a_W = -d_w\Delta y;\ a_N = -d_n\Delta y;\ a_S = -d_s\Delta y \quad [4.26]$$

4.2.1 Lid-Driven Cavity

Some of the applications of this discretization in the MATLAB code is shown in Fig. 4.2.2. The algorithm followed is SIMPLE algorithm that takes the Navier-Stokes equations through loops of many incorrect velocities and pressures until a specified tolerance limit is reached and the solution converges.

```
% incorrect u velocity
i=2:nx; j=2:ny+1;
  ut(i,j) = u(i,j)+dt*((-0.25)* ...
       (((u(i+1,j)+u(i,j)).^2-(u(i,j)+u(i-1,j)).^2)/dx ...
       +((u(i,j+1)+u(i,j)).*(v(i+1,j)+v(i,j)) ...
       -(u(i,j)+u(i,j-1)).*(v(i+1,j-1)+v(i,j-1)))/dy)...
       +(mu)*((u(i+1,j)-2*u(i,j)+u(i-1,j))/dx^2 ...
       +(u(i,j+1)-2*u(i,j)+u(i,j-1))/dy^2));
% incorrect v-velocity
i=2:nx+1; j=2:ny;
  vt(i,j)=v(i,j)+dt*((-0.25)*...
      (((u(i,j+1)+u(i,j)).*(v(i+1,j)+v(i,j))...
      -(u(i-1,j+1)+u(i-1,j)).*(v(i,j)+v(i-1,j)))/dx...
      +((v(i,j+1)+v(i,j)).^2-(v(i,j)+v(i,j-1)).^2)/dy)...
      +(mu)*((v(i+1,j)-2*v(i,j)+v(i-1,j))/dx^2 ...
      +(v(i,j+1)+-2*v(i,j)+v(i,j-1))/dy^2));
```

```
 %%% solving pressure poisson equation
pt = p;
for it = 1:maxit
  % pressure boundary conditions
  pt(1,:) = pt(2,:);
  pt(nx+2,:)= pt(nx+1,:);
  pt(:,1)= pt(:,2);
  pt(:,ny+2)= pt(:,ny+1);

  i=2:nx+1; j=2:ny+1;
     pt(i,j)=(0.5/(dx^2+dy^2))*((dy^2)*(pt(i+1,j)+pt(i-1,j))...
      +(dx^2)*(pt(i,j+1)+pt(i,j-1))...
      -(dx*dy/dt)*(dy*(ut(i,j)-ut(i-1,j))...
      +dx*(vt(i,j)-vt(i,j-1))));

    Er = max(max(pt-p));
    p = pt;
    if Er<10^-6
        break;
    end
end
%%% Time advancement
u(2:nx,2:ny+1)=...
         ut(2:nx,2:ny+1)-(dt/dx)*(p(3:nx+1,2:ny+1)-p(2:nx,2:ny+1));
v(2:nx+1,2:ny)=...
         vt(2:nx+1,2:ny)-(dt/dy)*(p(2:nx+1,3:ny+1)-p(2:nx+1,2:ny));

time = time + dt ;

uu(1:nx+1,1:ny+1) = 0.5*(u(1:nx+1,2:ny+2) + u(1:nx+1,1:ny+1));
vv(1:nx+1,1:ny+1) = 0.5*(v(2:nx+2,1:ny+1) + v(1:nx+1,1:ny+1));
pp = 0.5*(p(1:nx+1,1:ny+1) + p(2:nx+2,2:ny+2));
```

Fig. 4.2.2: Main parts of the MATLAB code for lid-driven cavity case

4.2.2 Flow Past a Static Cylinder

In this case also finite difference method, staggered grid configuration and SIMPLE algorithm has been used to solve the Navier-Stokes equation. Some of the important parts of the MATLAB code of this case is shown in Fig. 4.2.2.

```
%apply boundary conditions
u1(Ny/2-Ny/10:Ny/2+Ny/10,Nx/4-Nx/20:Nx/4+Nx/20) = 0.0;
v1(Ny/2-Ny/10:Ny/2+Ny/10,Nx/4-Nx/20:Nx/4+Nx/20) = 0.0;
%timestep value, relaxation factor, number of iterations (ENTER)
dt=0.1;
relaxation_factor=0.3;
total_iterations=1000;
residual_max = zeros(total_iterations,1);
%check CFL criteria (CHECK!)
CFL_x = max(max(u))*dt/dx;
CFL_y = max(max(u))*dt/dy;
```

Fig. 4.2.2: Definition of boundary conditions and relaxation factors, number of iterations and maximum residual in MATLAB code of flow past static square cylinder

4.2.3 Vortex-Induced Vibration (VIV) of flow past square cylinder

A similar case of flow past an elastically mounted square cylinder has been simulated using in-house MATLAB code. The most challenging part of the code is the change in grid points and thus boundary of the bluff body in the fluid domain at every time step. Thus, the boundary condition is accordingly adjusted as shown in Fig. 4.2.3.1. The structural dynamics have been solved using Newmark-ß method which is shown in Fig. 4.2.3.2.

```
%apply boundary conditions


% TIME LOOP STARTS HERE
dt = 0.1;

y = 0;
ydt = 0;
ydt2 = 0;
for t = 1:100
% %     t = (time - 1)*dt;

% VARIABLE BOUNDARY CONDITION FOR SQUARE CYLINDER
x_min = Nx/5 - Nx/20; x_max = Nx/5 + Nx/20;
y_min = Ny/2 - Ny/10; y_max = Ny/2 + Ny/10;
%y_min = Ny/2 - Ny/10 + int16(y(t)); y_max = Ny/2 + Ny/10 + int16(y(t));

u1(y_min:y_max,x_min:x_max) = 0.0;
v1(y_min:y_max,x_min:x_max) = 0.0;


%timestep value, relaxation factor, number of iterations (ENTER)

relaxation_factor = 1;
total_iterations=100;
residual_max = zeros(total_iterations,1);
%check CFL criteria (CHECK!)
CFL_x = max(max(u))*dt/dx;
CFL_y = max(max(u))*dt/dy;
```

Fig. 4.2.3.1: Time-varying boundary conditions to address change in position of cylinder subjected to vortex-induced vibrations

```
 Force = (sum(dx*p(y_min,x_min:x_max)) - sum(dx*p(y_max,x_min:x_max)));
 Force_save(t) = Force;
% % Newmark-Beta constants
%
 gamma = 0.5;
 beta = 0.25;
 dr = 5;   %damping ratio
 m = 50;     % mass of structure
 ws = 100;     % natural frequency of square
 c = 2*m*ws*(dr/100);
 k = m*ws^2;
 K = k + 3*c/dt + 6*m/(dt)^2;
 a = 6*m/dt + 3*c;
 b = 3*m + dt*c/2;
%
% % Newmark-Beta method
%
  deltaP=Force+a*ydt(t)+b*ydt2(t);
  deltay=deltaP/K;
  deltaydt=(gamma/beta/dt*deltay-gamma/beta*ydt(t)+dt*(1-gamma/beta/2)*ydt2(t));
  deltaydt2=(1/beta/(dt^2))*deltay - ydt(t)/beta/dt - ydt2(t)/2/beta;
  y(t+1) = y(t)+deltay;
  ydt(t+1)=ydt(t)+deltaydt;
  ydt2(t+1)=ydt2(t)+deltaydt2;
%
t
p_select = 1:1:100;
p_select = p(16,20);
P_alltime(t) = p_select;
```

Fig. 4.2.3.2: Newmark-ß method to address displacement of cylinder subjected to vortex-induced vibrations and thus update new displaced location in the code

4.2 VORTEX-INDUCED VIBRATION (VIV) OF 3D OFFSHORE SPAR PLATFORM

As offshore spar platforms are circular in cross-section, so the flow past an elastically mounted 3D circular cylinder is simulated to capture the vortex-induced motion. The problem statement is explained in detail in Section 5.3 of Chapter-5. This problem onwards the simulations have been carried out using C++ based open-source flow solver OpenFOAM (ver. 2212 in Ubuntu LTS

20.04), due to the fact that it is much more robust in nature than MATLAB. The number of cells used to discretize the entire domain in MATLAB was around 200, whereas the domain in OpenFOAM could be discretised to nearly 3.5 lakhs cells. OpenFOAM uses collocated type of grid configuration, that is, the parameters are stored at cell center and is a finite-volume based solver. The finite volume schemes used in this study are detailed in Annexure-II: Section-4.3. An extensive detail of the numerical methodologies available in OpenFOAM is presented in this chapter.

The main three directories generally present in any OpenFOAM tutorial case are

4.3.1) 0 4.3.2) constant, and 4.3.3) system.

The files and folders inside these folders are shown below:

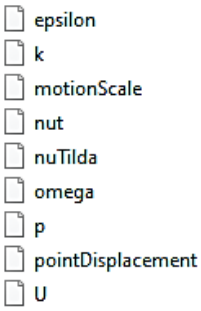


Fig. 4.3.1. Inside 0 directory

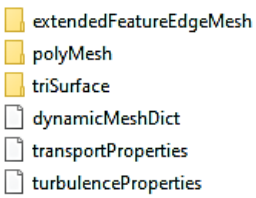


Fig. 4.3.2 Inside Constant directory

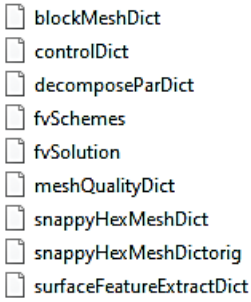


Fig. 4.3.3 Inside System directory

4.3.1) The 0 directory is to set all the initial and boundary conditions for free-stream velocity, pressure, turbulent kinetic energy, kinematic viscosity and many more.

4.3.2) The constant directory contains the file to input fluid properties and turbulence properties, if any. Along with these, the dynamicMeshDict is present here. The detailed explanation of this dictionary and its use is explained next.

4.3.2.1) dynamicMeshDict: The dynamicMeshDict is required when solvers that cause mesh motion are used. As a result, it manages how mesh morphs or deforms. The default base solver of any solver using this package has a "DyM" added to it. This is the dictionary that defines the two-way coupling.

dynamicMeshDict

The dynamicMeshDict used in the present work is shown in two figures, namely, Fig. 4.3.2.1a and Fig. 4.3.2.1b.

```
dynamicFvMesh   dynamicMotionSolverFvMesh;

motionSolverLibs (sixDoFRigidBodyMotion);

motionSolver    sixDoFRigidBodyMotion;
```
1

Fig. 4.3.2.1a: dynamicMeshDict

4.3.2.1.1) The dynamicFvMesh solver is particularly used for Fluid-Structure Interaction (FSI). It morphs the mesh around a specified set of boundaries and the meshing motion is calculated based on the pressures on those boundaries. On the other hand, dynamicMotionSolverFvMesh retrieves the feedback for the fluid simulation. It alters the velocity boundary conditions (U field) on the included borders to specify the local velocity of the stated body. This local velocity should take into account any associated translational and rotational motions. This mesh control is used to solve problems involving rigid body motion. The following areas are defined in dynamicMeshDict:

- Mesh morphing control
- Physical parameters of the rigid body
- Parameters to control how the 6DoF solver will actually solve the body motions
- Forces and motion constraints on the body, in addition to fluid forces.

The motion solver library is mentioned using 'motionSolverLibs' and the 'sixDoFRigidBodyMotion' motion solver is imported. This must be incorporated in any code definition for the dynamicMotionSolverFvMesh dictionary. The diffusivity parameter and inverseDistance are explained in Annexure-II: Section-4.3.2.

```
sixDoFRigidBodyMotionCoeffs   2
{
    patches         (cylinder);
    innerDistance   0.5;
    outerDistance   2.5;
```
2a

```
    centreOfMass    (0.0 0.0 1.0);

    // Density of the solid
    rhoSolid        2.0;

    // Cylinder mass
    mass            3.141592654;

    // Cylinder moment of inertia about the centre of mass
    momentOfInertia (1.243547092 1.243547092 0.392699082);    //no rotation
    g               (0 0 0);
        rho             rhoInf;
        rhoInf          1.0;
        report          on;
```
2b

```
report          on;
accelerationRelaxation 0.4;
```
2c

```
solver
{
    type Newmark;
}
```
2d

```
constraints
{
    fixedLine
    {
        sixDoFRigidBodyMotionConstraint line;
        direction (0 1 0);
    }

    noRotation
    {
      sixDoFRigidBodyMotionConstraint   orientation;
      centreOfRotation                  (0 0 1);
    }
}
```
2e

```
restraints
{
    verticalSpring
    {
        sixDoFRigidBodyMotionRestraint linearSpring;
        anchor          (0 0 0);
        refAttachmentPt (0 0.1 0);
        stiffness       3.45;
        damping         0;
        restLength      0.1;
    }
}
}

// ************************************************************
```
2f

Fig. 4.3.2.1b: dynamicMeshDict

4.3.2.1.2) sixDoFRigidBodyMotion Coefficients

Several measures related to structure motion are defined by these parameters. The application of each parameter can be categorized into several types:

4.3.2.1.2.1) Mesh Morphing Control: (**2a** from Fig. 4.3.2.1b) The solver specifies an innerDistance and outerDistance parameter that control the extent of mesh morphing.

- The mesh nodes move as a rigid body directly within the innerDistance.
- The mesh nodes are morphed between the innerDistance and outerDistance.
- Beyond the outerDistance, no morphing occurs.

4.3.2.1.2.2) Body Definition: (**2b** from Fig. 4.3.2.1b) The mass properties of the 6DoF body is defined. The ‘centreOfMass’ defines the centroid of the body at the initial position, in the global coordinate system. The ‘momentOfInertia’ defines the moment of the inertia of the body in the three principal axes (XX, YY, ZZ). The cross products of inertia are not defined. Both the ‘centreOfMass’ and ‘momentOfInertia’ are entered as vector lists. The mass applies to the entire

body. The density of solid body, density of fluid and mass is specified by ‘rhoSolid’, ‘rho’ and ‘mass’ respectively.

4.3.2.1.2.3) 6DoF Solver Control: (**2c** and **2d** from Fig. 4.3.2.1b)

2c) Acceleration Relaxation and Damping: These two parameters are to maintain stability of the 6DoF solver in situations of high acceleration. The high acceleration is fetched into the fluid domain which in turn creates high fluid forces, resulting in rapid deceleration. This leads to the diverging of 6DoF solver. OpenFOAM provides two parameters to avoid this diversion.

AccelerationDamping is the used to eliminate divergence from sudden acceleration. The calculated acceleration on the body is reduced by a certain proportion to the magnitude of the acceleration, similar to a damping coefficient. This results in sudden accelerations receiving large amounts of relaxation and normal accelerations, that are typical for the result of the time history receiving relatively less acceleration. The parameter can range from 0.0 to 1.0.

AccelerationRelaxation is a direct reduction on the acceleration. The acceleration is reduced to the new acceleration by multiplying with the relaxation factor. This is then applied to the newly derived boundary conditions and body velocity. The parameter can range from 0.0 to 1.0. Too much difference between calculated fluid forces and resulting motion shall lead to divergence.

2d) Solver Type: The use of Newmark-Beta method, Crank-Nicholson method or symplectic method is detailed in Annexure-II: Section-4.3.2.

4.3.2.1.4) Forces Definitions: (**2e** from Fig. 4.3.2.1b) The restraints act as reactionary forces other than the fluid interaction. Fluid interaction is automatically included whenever a patch is specified as part of the body. The restraint forces model other forces through simple reaction force terms. The exact force magnitude will change depending on body behavior. There can be multiple force types, that is, restraint conditions in this library.

In addition to the fluid interaction, the constraints function as reactive forces. Every time a patch is designated as a component of the body, fluid interaction is automatically taken into account. Through straightforward reaction force concepts, the restraint forces represent other forces. Depending on how the body responds, the precise force magnitude will differ. Multiple force kinds, or constraint conditions, can exist in this library.

Force type keywords available in OpenFOAM are:

- constantForce
- linearDamper
- sphericalAngularDamper

- *linearSpring*
- linearAxialAngularSpring
- tabluatedAxialAngularSpring

Out of these, linearSpring has been used in the present work. The remaining types are explained in Annexure-II: Section-4.3.2.

linearSpring: Fig. 4.3.2.1b **(2f)**

For restraint forces, the linearSpring offers a more complete model. The following are the main characteristics of this model:

- force related to body displacement; stiffness.
- damping, a force correlated to body velocity.
- attachmentPoint, which allows for connection between body forces and body moments. The coupling is determined by the separation between the body centre of gravity and refAttachmentPt.
- restLength, which allows to choose the spring's initial settings of compression or tension.

Anchor Position

Two end points serve as the spring's definition. The fixed end point is specified by the anchor. Throughout the simulation, this will stay in its fixed location.

Body Attachment Position

This describes the spring's second terminus. The body's point of connection is here. There will be forces and paired moments produced by the spring. The moment is produced by the moment arm between the refAttachmentPt and the body's centre of gravity.

Stiffness

The linearSpring forces' spring component is represented by this stiffness. Linear deflection and force vary in a linear relationship. The part of the force that reacts to body displacements is described here. The displacement at the refAttachmentPt is the basis for the forces. This covers the body's translational and rotary motions. Units of N/m are used to measure stiffness. This reactive force can alternatively be defined as having zero stiffness. This will result in an object that only functions as a damper.

Damping

The linearSpring forces' damper component is this damping. Linear velocity and force vary in a linear relationship. This details the force's component that reacts to body velocity. Based on the velocity at the refAttachmentPt, the forces are applied. This covers the body's translational and rotary motions. Ns/m units are used to specify damping. A damping value of 0 defines for an undamped system.

Resting Length

The resting length is used to describe the spring's initial state. The linear distance at which the spring applies no force is known as the resting length.

4.3.2.1.5) Motion Definitions: A set of restricted motions is referred to as constraints. The permissible movements of the body are directly controlled by constraints. These don't apply reactive forces like restraints do. There is no force created by constraints. They only limit the body's movements. A collection of established constraints makes up the constraints parameters. To limit body motions in various ways, one can construct combinations of different constraint kinds.

List of MotionType keywords

- axis
- *line*: The line constraint concentrates on linear motion. The direction of rotation is unrestricted. Only linear motion along the vector indicated by the direction keyword is allowed by the line constraint. This restricts the range of possible body motions to one degree of freedom for linear motion and three degrees for rotational motion. A vector line is used to define the direction. The direction definition is automatically scaled by OpenFOAM so that it can be understood as a unit vector definition.
- *plane*: The plane limitation allows for 2D linear motion but prevents 3D motion of the body. The plane's orientation determines the motion's direction. Anywhere inside the designated plane, the body is allowed to move linearly. The normal vector, which is defined by the "normal" keyword, defines the plane. The normal vector specification is automatically scaled by OpenFOAM so that it can be understood as a unit vector definition. The body's initial centre of gravity is assumed to be at the vector's origin when defining this plane constraint.
- Point
- orientation: The body is permitted unrestricted linear motion, but the orientation constraint prevents any rotational motion. The body's centre of mass is used if the "centreOfRotation" is not specified.

4.3.2.1.6) Output Control: (**2c** from Fig. 4.3.2.1b) The 6DoF body status is written to the time files by the "report" argument. The file sixDoFRigidBodyMotionState is contained within the folders of every time step. It contains the information of centre of rotation of the body at that specific time step.

The dynamicMeshDict of 2DOF VIV of cylinder is shown in Fig. 4.3.2.1b. The remaining features of dynamicMeshDict that is not used in the present work, but can be used in OpenFOAM are detailed in Annexure-II:Section-4.3.2.

4.3.3) *system* folder: This folder in OpenFOAM consists the following dictionaries:

- fvSchemes: for various finite volume schemes
- fvSolution: solution methodology
- controlDict: solver to be used, time step, Courant number
- surfaceFeatureExtract: to extract the structure into the fluid domain
- snappyHexMesh: to snap the structural mesh into the fluid domain mesh. This is detailed in Section-5.2 of Chapter-5.

4.3.3) system folder: This folder contains the input files for finite volume schemes, meshing controls, time control and the solver choice that needs to be used. The schemes for discretization and computing of all the terms of Navier-Stokes equation that are available in OpenFOAM are detailed in Annexure-II: Section-4.3.3.

4.3 LES ON FLOW PAST STATIC SQUARE CYLINDER AT REYNOLDS NUMBER 10^5

This case has been solved in OpenFOAM at a very high Reynolds number of 10^5 in order to and is presented in Section 3.3. As this is a transient simulation, like that of Section 4.3, the algorithm used to solve Navier-Stokes equation is pressure-implicit split operator (PISO) algorithm. The various types of algorithms to be used can be specified in /system/fvSolution and /system/controlDict. The pressure-implicit split operator (PISO), semi-implicit technique for pressure-linked equations (SIMPLE), or a combination PIMPLE algorithm are used by the majority of fluid dynamics solver applications in OpenFOAM. These algorithms, PISO and PIMPLE for transient issues and SIMPLE for steady-state, are iterative methods for linking equations for momentum and mass conservation.

Both procedures solve a pressure equation in a single time step, or solution step, with an explicit correction to velocity to meet momentum conservation. The momentum equation, or so-called momentum predictor, is optionally solved at the start of each phase.

The algorithms mostly differ in how they loop over the governing equations, even though they all solve the equations (although in different ways). The following list of input parameters governs the looping. They are entered into a dictionary with the algorithm's name, such as SIMPLE, PISO, or PIMPLE.

- nCorrectors: used by PISO, and PIMPLE, sets the number of times the algorithm solves the pressure equation and momentum corrector in each step; typically set to 2 or 3.
- nNonOrthogonalCorrectors: used by all algorithms, specifies repeated solutions of the pressure equation, used to update the explicit non-orthogonal correction, of the Laplacian term $\nabla.((1/A)\nabla p)$; typically set to 0 (particularly for steady-state) or 1.
- nOuterCorrectors: used by PIMPLE, it enables looping over the entire system of equations within on time step, representing the total number of times the system is solved; must be ≥ 1 and is typically set to 1, replicating the PISO algorithm.
- momentumPredictor: switch those control solving of the momentum predictor; typically set to off for some flows, including low Reynolds number and multiphase.

The problem statements of the works done is discussed in Sectio-5.1 of Chapter-5 and the results obtained with relevant discussions are detailed in Section-5.2 of Chapter-5.

CHAPTER 5: RESULTS AND DISCUSSIONS

5.1 INTRODUCTION

This section provides a comprehensive understanding of canonical flow problems using MATLAB and the vortex-induced vibration of offshore spar platforms using OpenFOAM. These numerical methodologies offer valuable insights into the behaviour of fluid flows and the dynamic response of structures, aiding in the design, optimization, and safety of engineering systems. By analysing and interpreting the obtained results, one can uncover the underlying mechanisms, assess the accuracy of the numerical models, and contribute to advancements in fluid mechanics and offshore engineering.

The research work dealt in this thesis has been divided into major 3 sub-parts. The first part consists of the in-house MATLAB coding done to solve the Navier-Stokes equation by Finite Difference Method. The second part consists of simulating the vortex-induced vibration of a 3D offshore spar platform using a C++ based open-source software, namely OpenFOAM, a Finite-Volume based solver. And the final part consists of the Large Eddy Simulations performed on a static square cylinder and subsequently reduce response with the help of a coupled Finite Volume-Finite Element procedure.

5.2 CANONICAL FLOW PROBLEMS USING IN-HOUSE MATLAB CODING

Using Semi-Implicit Method for Pressure Linked Equations (SIMPLE) continuity and Navier-Stokes equations are solved, using a pressure corrector. This technique is used to solve the following 2D cases:

5.2.1 Lid-driven cavity case
5.2.2 Flow past static square cylinder case
5.2.3 Flow past elastically mounted square cylinder case

5.2.1 Lid-driven cavity

5.2.1.1 Problem Statement

A square cavity with a lid at top, is taken into consideration. The square cavity side is taken to be 1m. and the walls are in no-slip condition. Initially the pressure inside and outside the cavity has been considered same. On removing the infinitely long lid in the direction as shown in Fig. 5.2.1.1, the fluid inside the cavity experiences changes in its velocity and pressure. This change has been studied through the 2D numerical analysis in MATLAB and validated with the results of [78]. The

Reynolds number considered are: 100, 400 and 1000. The domain has been divided into a 32x32 grid and the iterations have been carried on for a time step of 0.001 s. SIMPLE Algorithm has been used to solve this problem, for Reynold's Number (Re) 100, 400 and 1000. The velocity of the lid, U ('Ut' in Fig. 5.2.1.1) has been taken as 1 m/s, 4 m/s and 10 m/s for Re values 100, 400 and 1000 respectively.

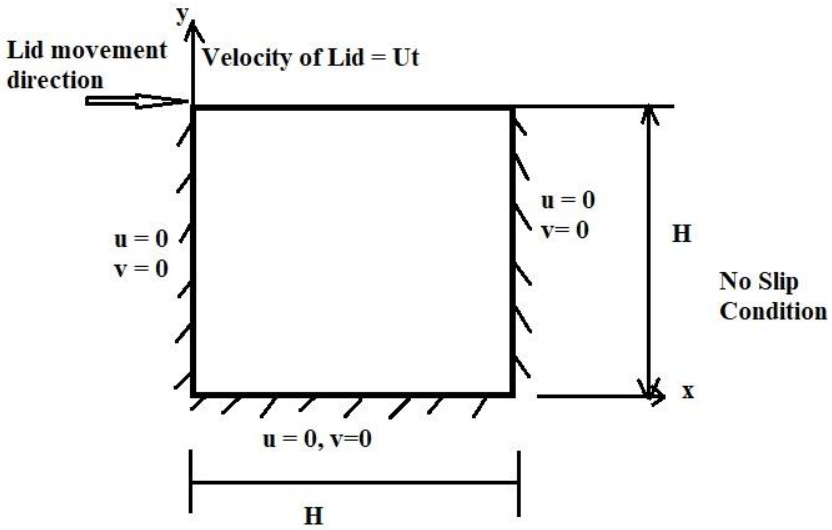


Fig. 5.2.1.1: Lid-driven cavity considered in problem 5.2.1.1 [2D, in-house MATLAB coding]

5.2.1.2 Results and Discussions

An attempt has been made to validate the velocity component contour and pressure contour due to the changes that the fluid inside the cavity experiences. This change has been studied through the numerical analysis in MATLAB and validated with the results of [78]. The Reynolds number considered are: 100, 400 and 1000.

- ***u and v velocity contour, pressure contour for Re = 100, Ut = 100 m/s***

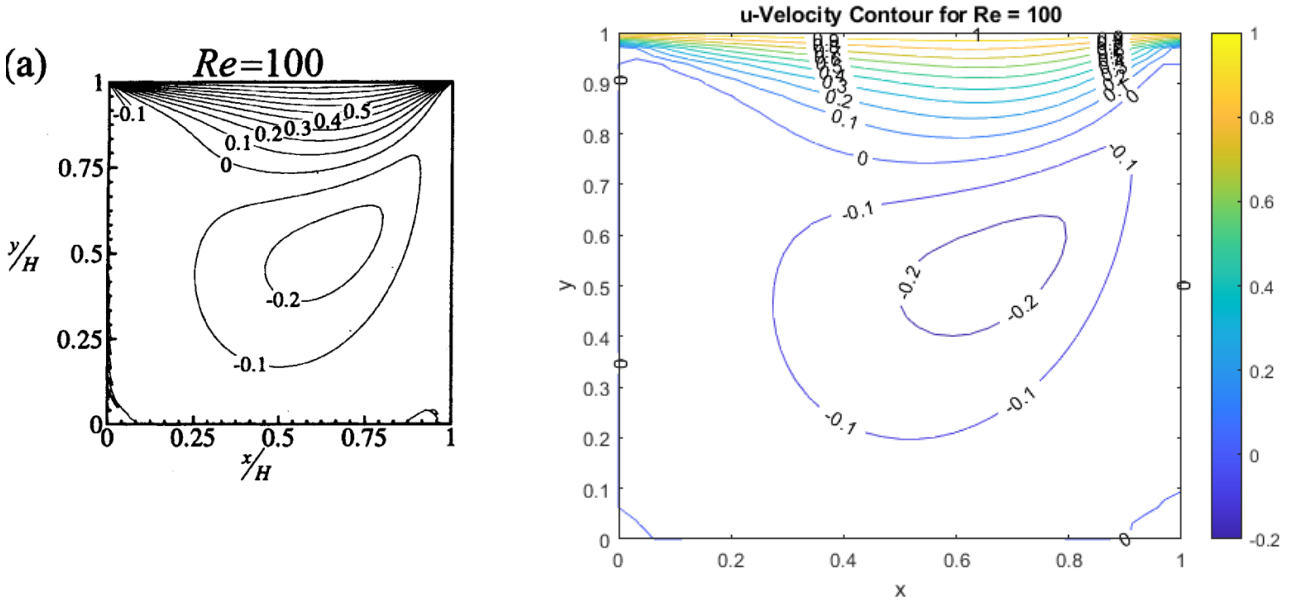


Fig. 5.2.1.2.a: **u velocity** contour for **Re = 100** using SIMPLE Method
(left: result mentioned in [78]; right: computed result) [2D, in-house MATLAB coding]

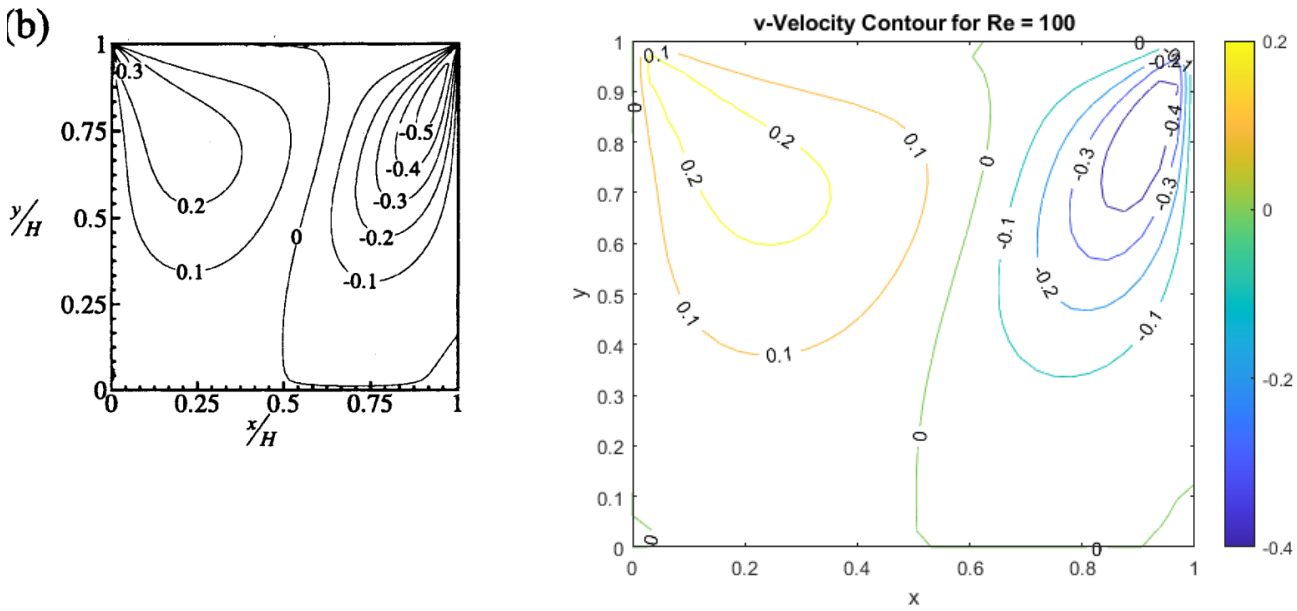


Fig.5.2.1.2.b: **v velocity** contour for **Re = 100** using SIMPLE Method
(left: result mentioned in paper; right: computed result) [2D, in-house MATLAB coding]

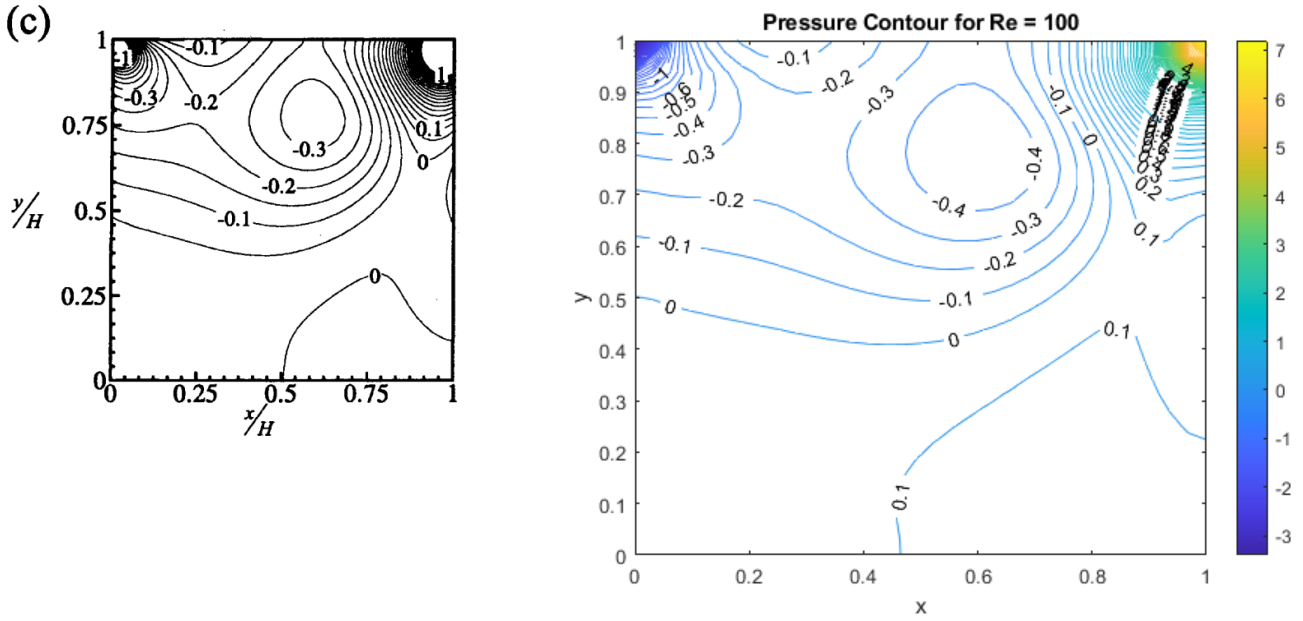


Fig. 5.2.1.2.c: **Pressure** contour for **Re = 100** using SIMPLE Method
(left: result mentioned in paper; right: computed result) [2D, in-house MATLAB coding]

- ***u and v velocity contour, pressure contour for Re = 400, Ut = 4 m/s.***

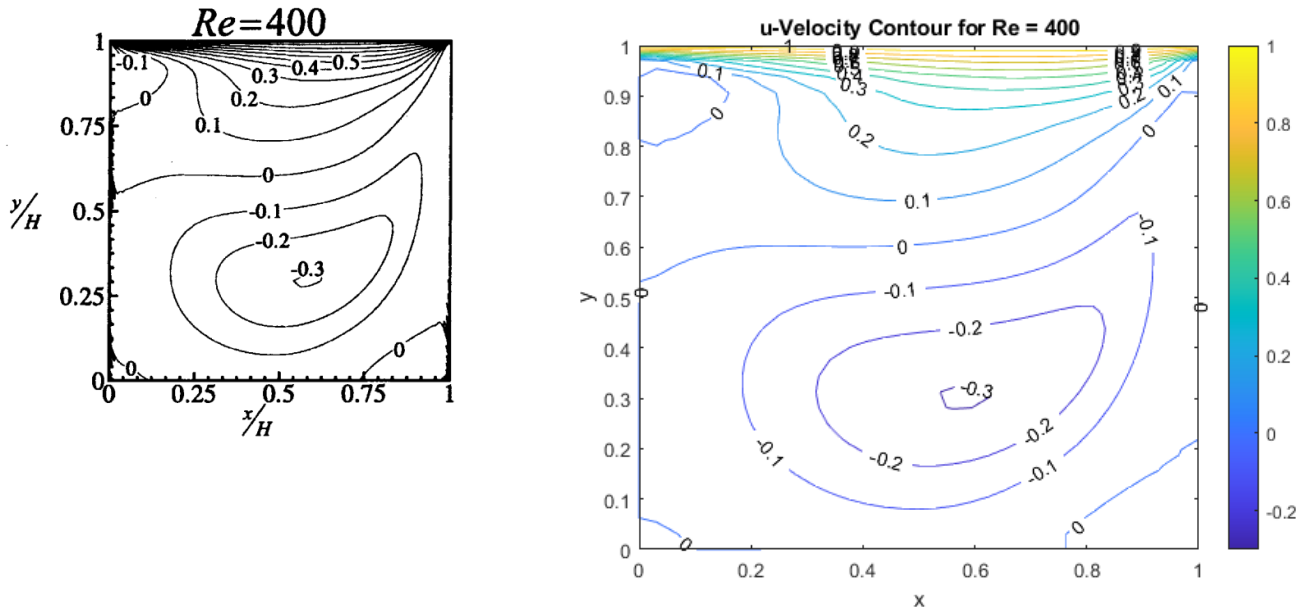


Fig. 5.2.1.2.d: **u velocity** contour for **Re = 400** using SIMPLE Method
(left: result mentioned in [78]; right: computed result) [2D, in-house MATLAB coding]

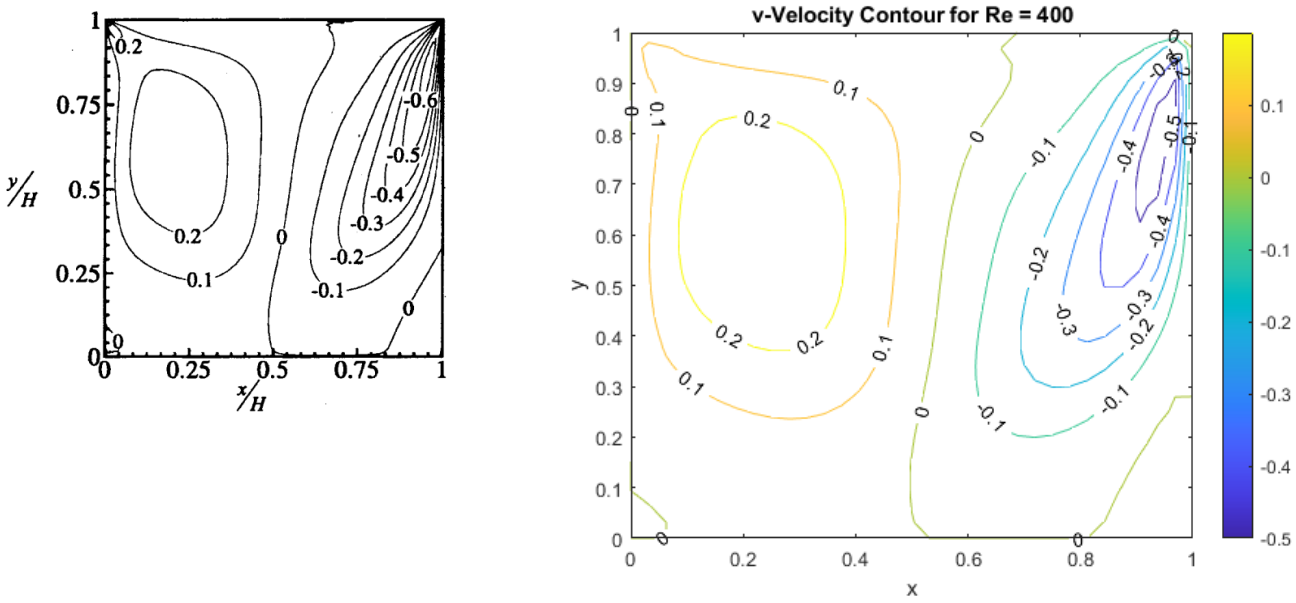


Fig. 5.2.1.2.e: **v velocity** contour for **Re = 400** using SIMPLE Method
(left: result mentioned in [78]; right: computed result) [2D, in-house MATLAB coding]

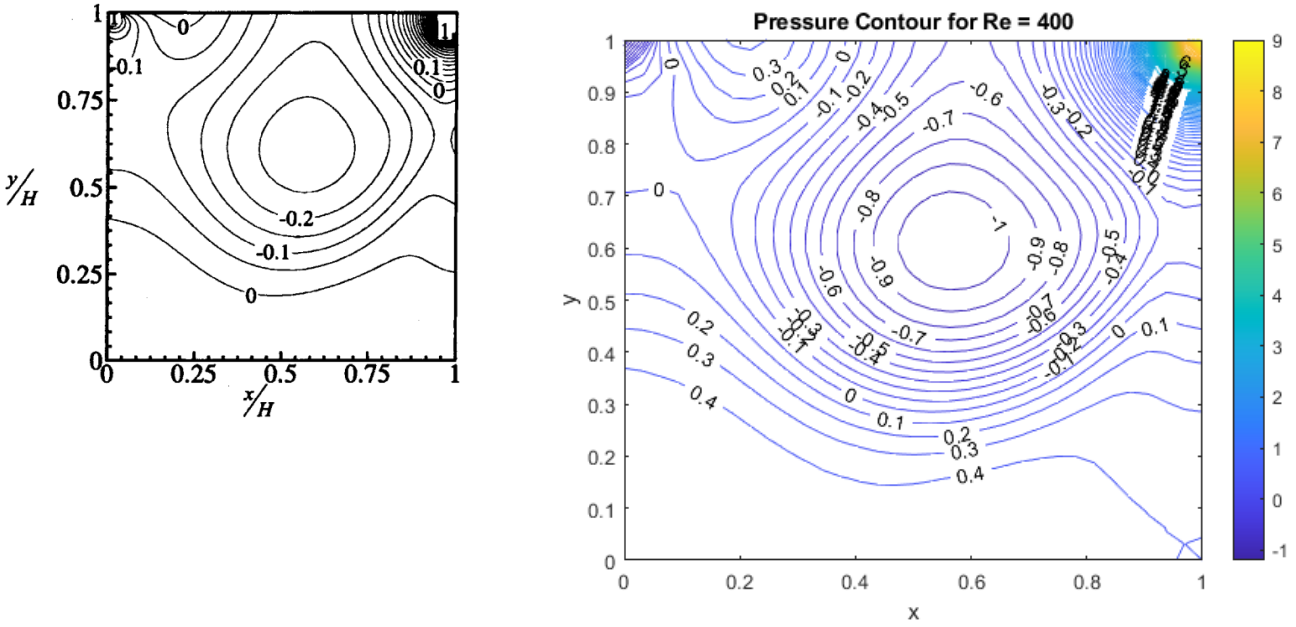


Fig. 5.2.1.2.f: **Pressure** contour for **Re = 400** using SIMPLE Method
(left: result mentioned in [78]; right: computed result) [2D, in-house MATLAB coding]

- ***u and v velocity contour, pressure contour for Re = 1000, Ut = 10 m/s***

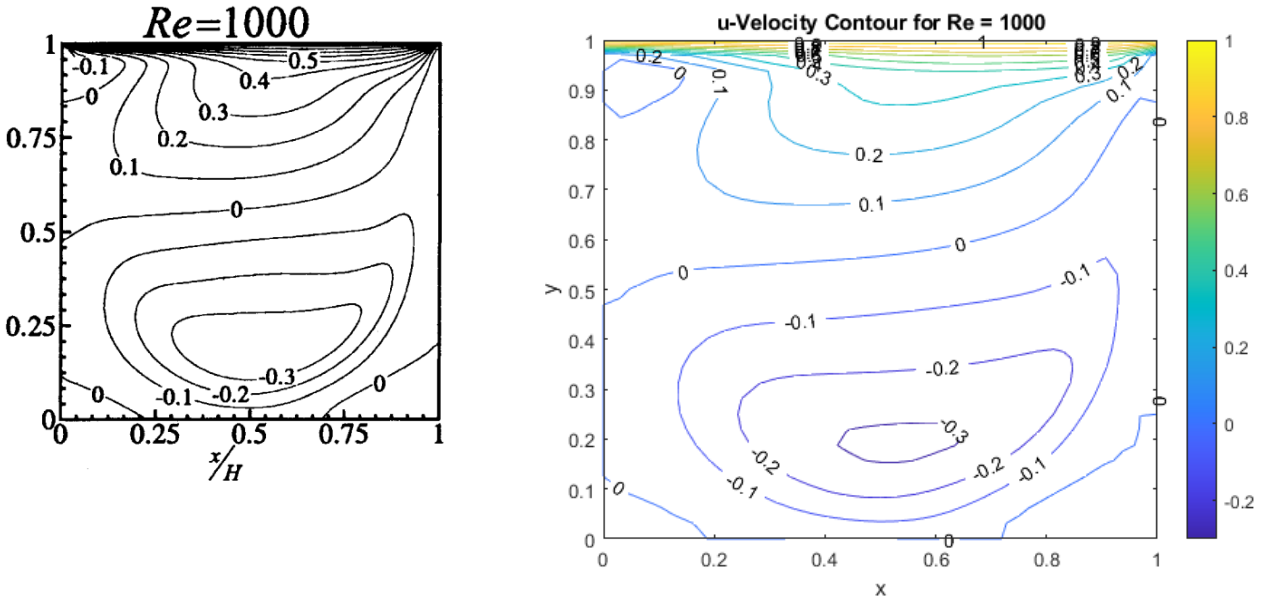


Fig. 5.2.1.2.g: **u** contour for **Re = 1000** using SIMPLE Method
(left: result mentioned in [78]; right: computed result) [2D, in-house MATLAB coding]

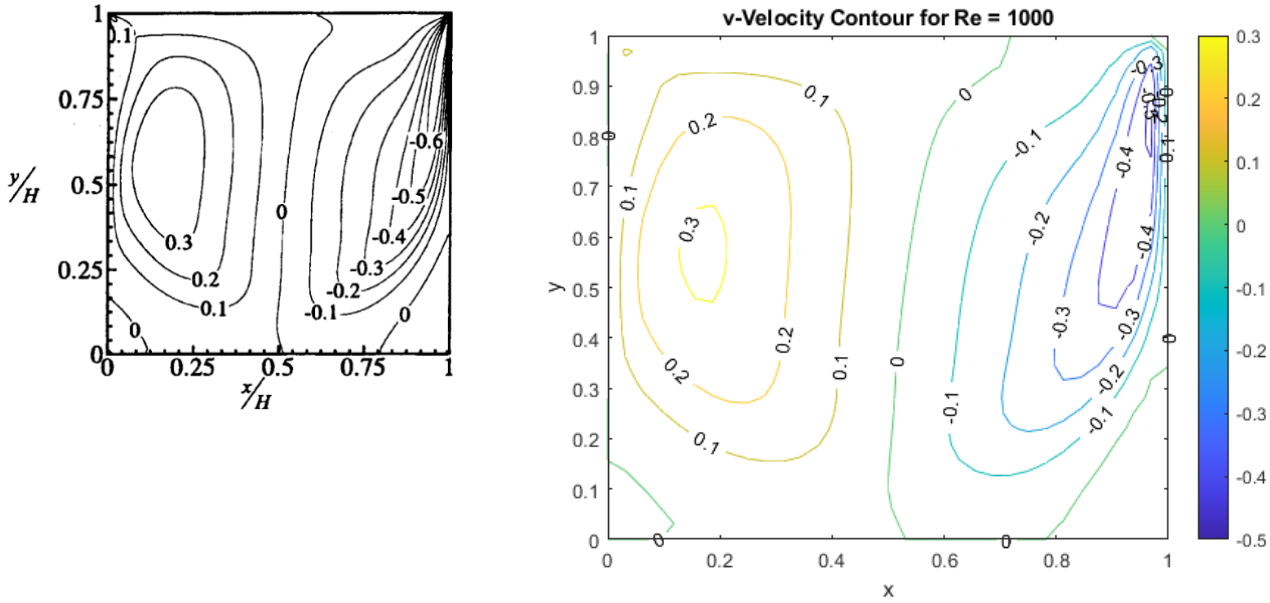


Fig. 5.2.1.2.h: **v velocity** contour for **Re = 1000** using SIMPLE Method
(left: result mentioned in [78]; right: computed result) [2D, in-house MATLAB coding]

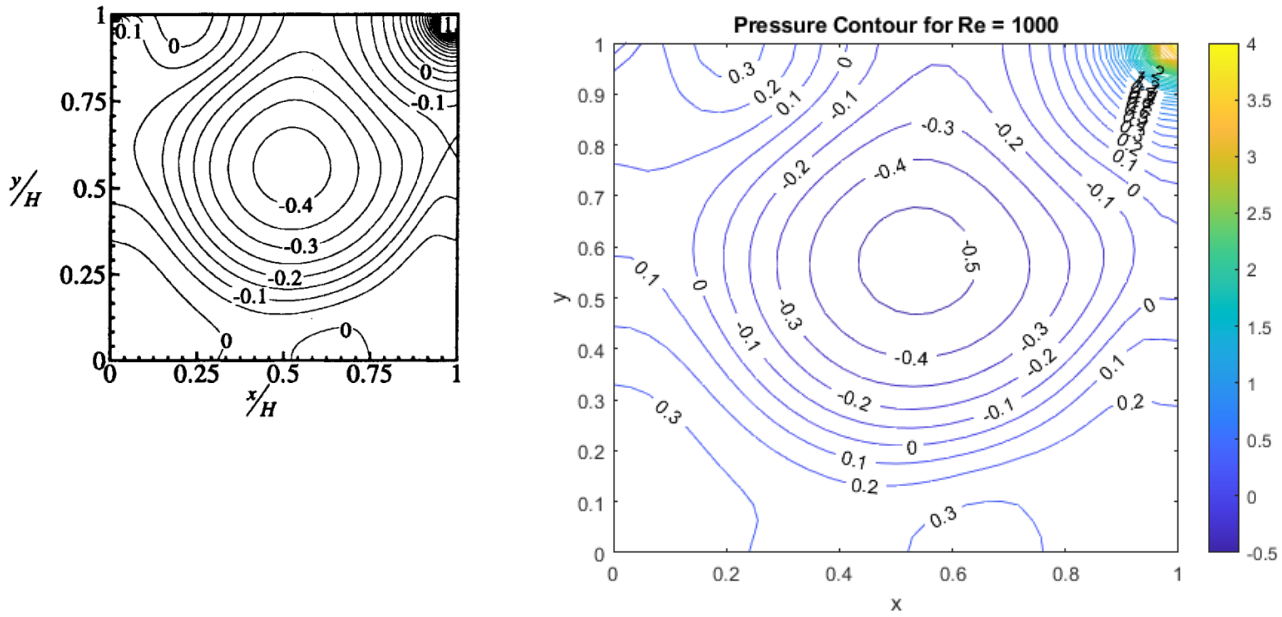


Fig. 5.2.1.2.i: **Pressure** contour for **Re = 1000** using SIMPLE Method
(left: result mentioned in [78]; right: computed result) [2D, in-house MATLAB coding]

5.2.1.3 Conclusions

The present numerical code is able to perfectly solve the Navier Stokes equations and depict the flow physics. It is validated with the reported analytical works. It can solve the u, v and p which in turn leads to compute other variation of the problems like flow over fixed or movivable square cylinder, or structures with curved surface.

5.2.2 Flow Past Static Square Cylinder

5.2.2.1 Problem Statement

A laminar flow is considered in a 2D domain where a square cylinder is placed. The domain is 10 m. along x-axis and 5m. along y- axis originates at O. The square cylinder is of side 1 m. placed within A (2,2); B (3,2); C (3,3); D (2,3) of the chosen domain. The flow is parallel to x-axis.

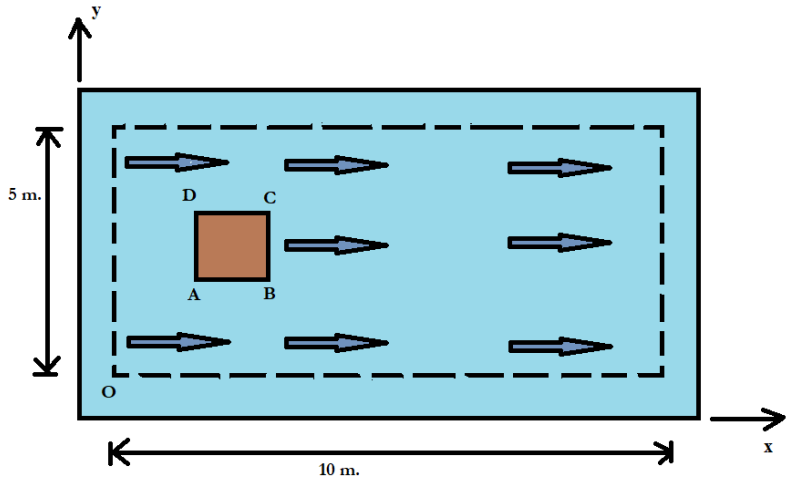


Fig. 5.2.2.1: Schematic of flow past square cylinder case study [2D, in-house MATLAB coding]

The domain has been divided into a 100x50 grid. SIMPLE Algorithm has been used to solve this problem. The velocity of the fluid flow is 1 m/s. The pressure contour for such a flow domain and the obstruction is found out using MATLAB and compared with the results of ANSYS-Fluent. The free-stream velocity at the inlet (U) is 1 m/s.

5.2.2.2 Results and Discussions

The pressure contour for the flow past static square cylinder computed in MATLAB and ANSYS Fluent are shown in Fig. 5.2.2.2.a and Fig. 5.2.2.2.b respectively.

5.2.2.3 Conclusions

From the results, it can be concluded that the in-house MATLAB code gives results which are in very good agreement with that when computed using a standard commercial software (ANSYS Fluent v.14.5). Thus, this MATLAB code, boundary conditions of which is shown in Fig. 4.2.2 can be used for flow past any static bluff body completely immersed in the fluid.

Further, this in-house MATLAB code has been extended to observe vortex-induced vibration of a square cylinder allowed to oscillate in the cross-flow direction.

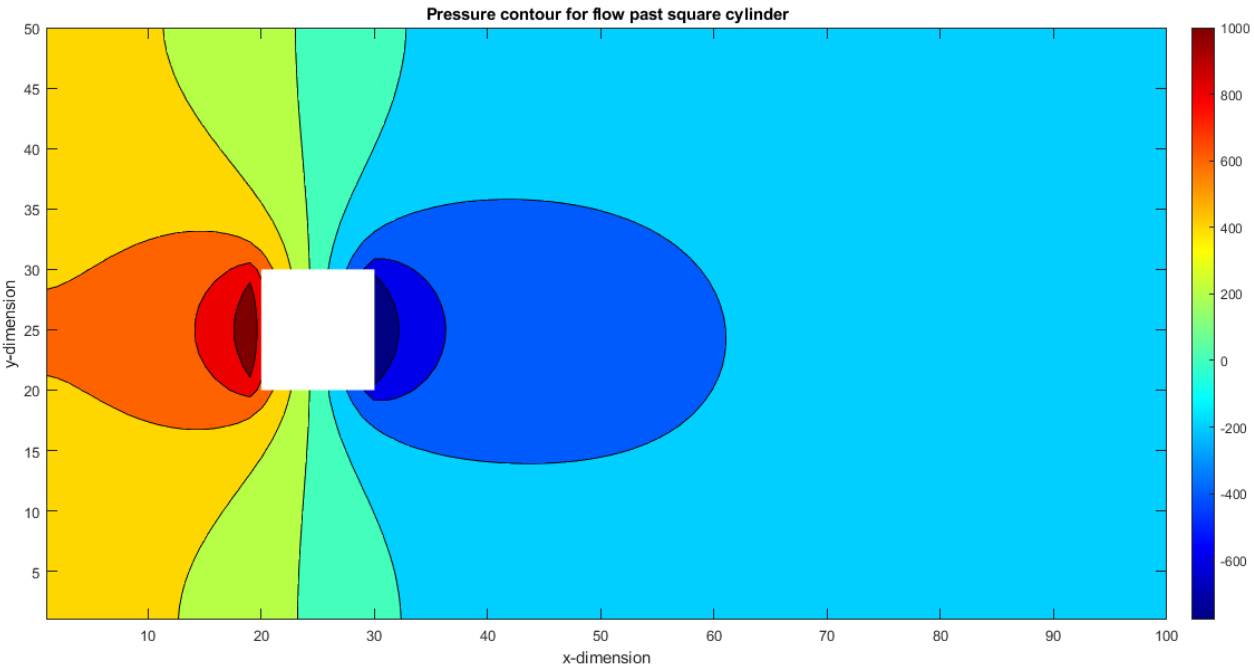


Fig.5.2.2.2.a: Pressure contour for flow past static 1m. x 1m. square cylinder obtained using MATLAB; U = 1 m/s, along x-direction
[2D, in-house MATLAB coding]

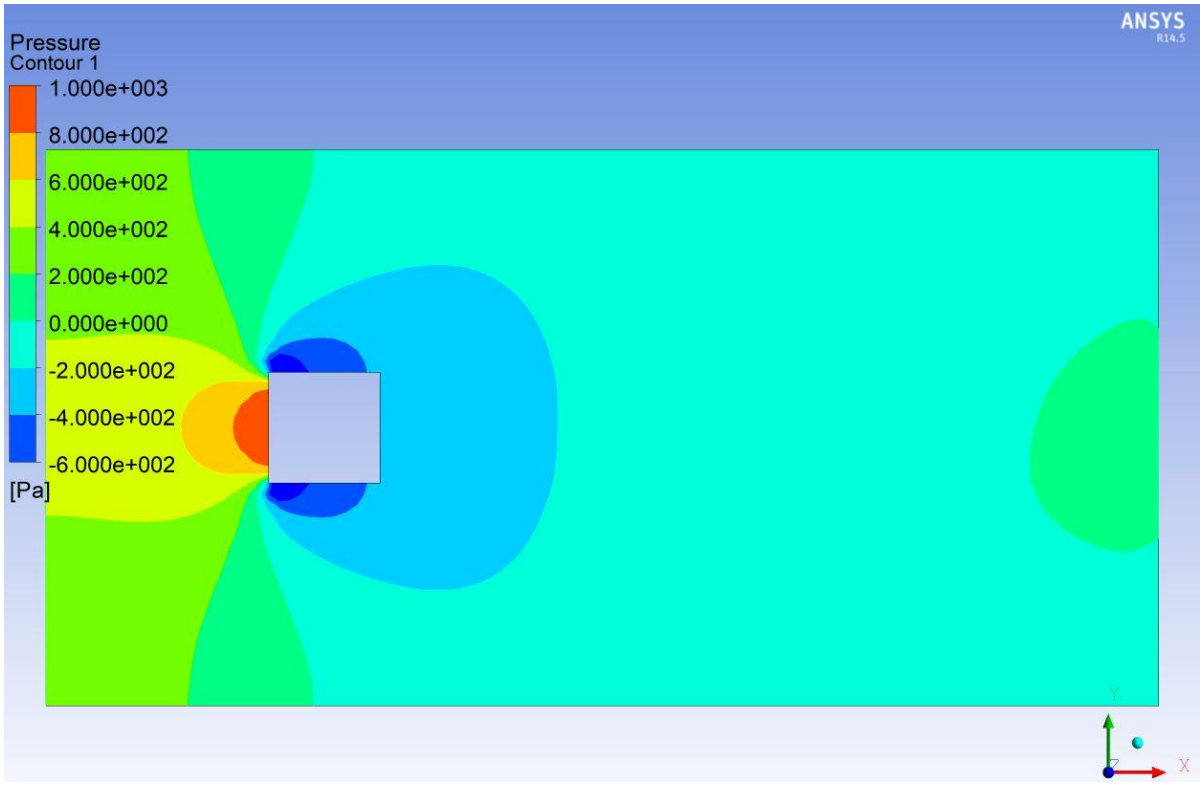


Fig. 5.2.2.2.b: Pressure contour for flow past square cylinder obtained using ANSYS-FLUENT
[2D, in-house MATLAB coding]

5.2.3 Vortex-Induced Vibration of Flow Past Square Cylinder

5.2.3.1 Problem Statement

The in-house MATLAB code used to simulate 5.2.2 has been extended to observe vortex-induced vibration of the square cylinder allowed to oscillate in the cross-flow direction (y-direction). This is also a two-dimensional (2D) case. The only difference from 5.2.2 is that the cylinder system contains a particular stiffness, that is, it is elastically mounted in the cross-flow direction, thus it is a transient simulation.

5.2.3.2 Results and Discussions

The in-house MATLAB code to solve this problem, applying Newmark-β method in order to update the displaced position of cylinder at every time step is shown in Fig. 4.2.3.1 and 4.2.3.2. The pressure-velocity field around the cylinder at two different time steps are presented in Fig. 5.2.3.2.

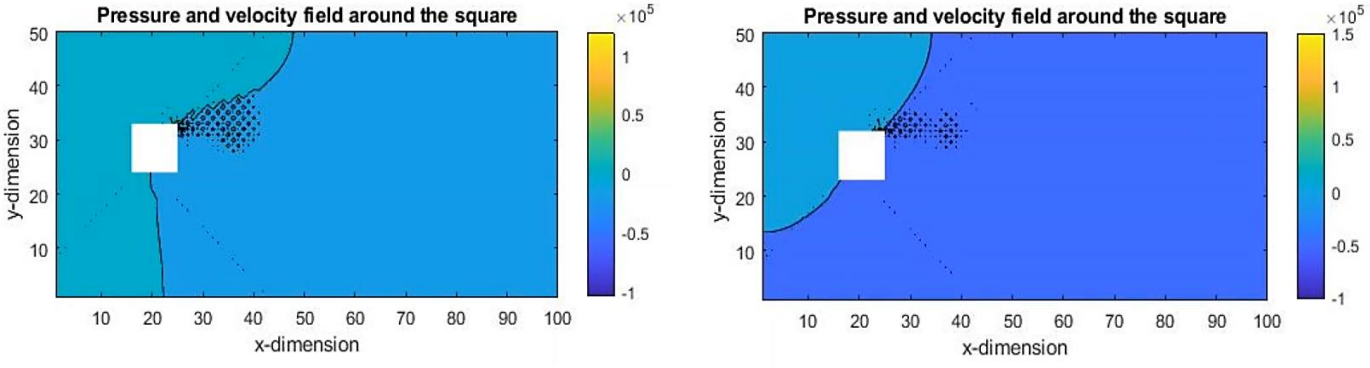


Fig. 5.2.3.2: Pressure-velocity field around the elastically mounted (along cross-flow direction) 1m. x 1m. square cylinder for two different consecutive time steps, U = 1 m/s, along x-direction [2D, in-house MATLAB coding]

5.2.3.3 Conclusions

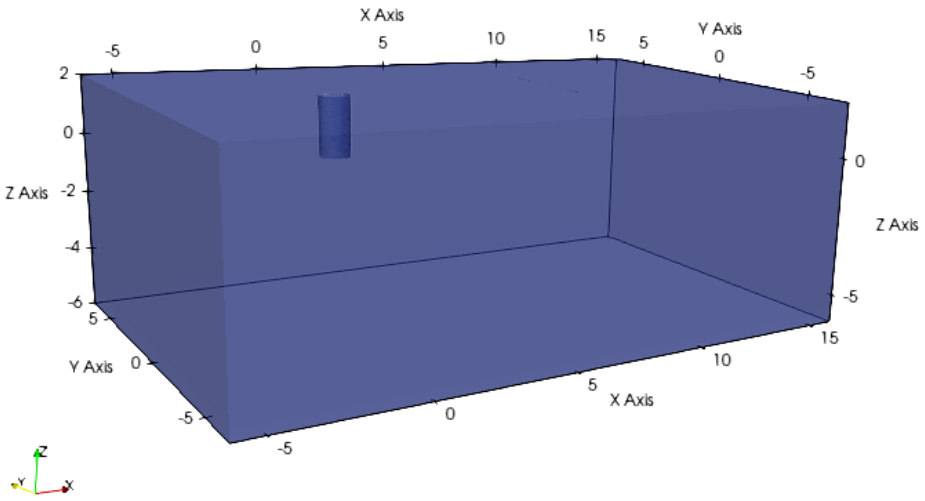
It can be seen in Fig. 5.2.3.2 that there is change in position of the cylinder in the two consecutive time steps. So, it can be concluded that the understanding of the flow physics and solving of Navier-Stokes equation for flow past bluff body is satisfactory and the problem can be extended to simulate 3D VIV of bluff bodies. In MATLAB the fluid mesh is very coarse where staggered grid configuration is used to solve Navier-Stokes equation by finite difference method and thus to solve for a much finer mesh, the next simulations are carried out in finite-volume based OpenFOAM (ver. 2212 in Windows Subsystem for Linux) where collocated grid configuration is used. Point of separation in flow past square cylinders is easy to detect, so a more realistic circular cross-section is considered next [9].

5.3 VORTEX-INDUCED VIBRATIONS OF 3D OFFSHORE SPAR PLATFORM IN OPENFOAM

5.3.1 Problem Statement

The interaction between fluid and a structure immersed in it is studied in this work. The structure considered in this work is an offshore spar platform in the form of a 3D circular cylinder of aspect ratio (L/D) 2.

The geometry of the circular cylinder with an aspect ratio[4] (L/D) of 2 studied in this work is illustrated in Fig. 5.3.2.2. This geometry is constructed using the open-source software Salome. The selection of this aspect ratio is on par with the OC4 DeepCwind semi-submersible platform[53], thus, a relatively short cylinder with a low aspect ratio of L/D = 2 is employed. A low mass ratio of m^* of 2 is adopted, which is the ratio of the mass of the structure and mass of the displaced fluid. The mass of the displaced fluid is calculated from the fluid density and volume of the cylinder. The aspect ratio L/D and mass ratio m* opted in this study are the same as those of [20] and [52] for a finite cylinder without the base column. The cylinder is subjected to uniform flow (U) with a free-stream velocity of 1 m/s. The Reynolds number is taken to be 300, where the diameter of the cylinder (D) is taken to be 1m., free-stream velocity as in the work of [20]. The Reynolds number at 300 is chosen since it is low enough to directly solve the Navier-Stokes equations avoiding the potential uncertainties involved with the use of turbulent models and on the other hand it is already in the turbulent regime. The fluid considered here is Newtonian[5].



[4] where L is the overall length of the cylinder and D is its characteristic diameter

[5] A Newtonian fluid is one that has a constant viscosity and a shear rate that is directly proportional to the shear stress.

Fig. 5.3.1: Numerical domain of the present study

This study is done for:

5.3.1a) Spring study: The position, arrangement and number of springs to be used to suffice the stiffness of the system is found out.

5.3.1b) Cylinder allowed to move only in cross-flow direction (1DOF)

5.3.1c) Cylinder allowed to move only in cross-flow as well inline direction (2DOF) for the reduced velocity where maximum amplitude is obtained as per 5.3.1b.

For this problem, the pimpleFoam solver with a moving mesh capability is selected, where a PIMPLE (merged PISO-SIMPLE) algorithm is applied to deal with the coupling of velocity and pressure in a segregated way in OpenFOAM.

5.3.2 Results and discussions

[***This is the most important segment of the present work, where a 3D VIV model for offshore spar platform is developed using OpenFOAM, and the flow-induced motion of the structure is studied.***]

The interaction between fluid and a structure immersed in it is studied in this work. The structure considered in this work is an offshore spar platform in the form of a circular cylinder of aspect ratio (L/D) 2. The entire work of this section has been done in OpenFOAM. A transient, first-order implicit, bounded scheme is used for temporal discretisation while an unbounded Gaussian second-order upwind scheme is adopted for convective terms. Cell-based linear interpolation scheme is used to interpolate cell centre values to face centres. Gradient terms are computed by a second-order, cell limited Gauss linear scheme.

5.3.2.1 Domain, grid and time-step independence study

The grid independence study is done in order to check whether the mesh chosen for the study is adequate to give results nearest to the experimental results. Prior to this, a domain independence study is carries out.

5.3.2.1.1 Domain Independence Study

This is mainly done to reduce the computational expenses. The width and length of the domain used in [20] is $40D \times 50D$ and the cylinder being placed 20D and 30D from the inlet and outlet boundary, respectively. The free end or bottom of the cylinder is at 10D distance from the bottom of the domain. The bottom of the domain is considered a wall, in order to represent it as a sea or ocean bed.

For the present study, a domain of width 14D and length 22D is chosen, where the cylinder is located at 6D from inlet boundary and 16D from outlet boundary. It is placed midway along the width of the domain and is at a distance 6D from the bottom of the domain, as shown in Fig. 5.3.1. The origin of the Cartesian coordinate system is located at the centre of the bottom, free-end of the cylinder.

From the domain independence study, it has been found that the chosen domain is adequate to carry out the study. Although it had been found in [79] that, if the distance between the bottom wall of the domain and bottom free-end of the cylinder is 20D or more, the effects of the walls on the VIV were negligibly small. From the domain chosen in this study and the comparison study given in Table-5.3.1, it can be clearly stated that the distance of 6D between these two walls have almost no effect on the VIV. The amplitude ratio, denoted as A_y/D, is the ratio of the amplitude of cylinder in cross-flow direction to its diameter.

Table 5.3.1: Comparison between the domains for reduced velocity (V_r) 6. L_x: Distance of cylinder center from inlet (along X); L_y: Distance of cylinder center from front or back of domain (along Y); L_z: Distance between bottom of cylinder and domain (along Z)

Domains	L_x	L_y	L_z	Amplitude (y-direction) to diameter ratio, A_y/D	RMS Lift Coefficient, CL_{rms}
Domain-1[20]	20D	20D	10D	0.75403	0.41768
Domain-2 (Present study)	6D	6D	6D	0.760243	0.43002

5.3.2.1.2 Grid Independence Study

The dependency on the computational mesh is very crucial. As in CFD, the variables vary linearly across the cells, so the cell size plays an important role in the computation of the parameters. But it is not possible to make a domain having infinite number of cells, so a particular mesh is chosen finer to which the CFD results do not vary much and provide almost accurate results. This is why mesh dependency study is done.

The meshing of this case is a very important part. For this transient simulation, the structure points in the fluid domain changes position at every time step and thus, the mesh is updated at every time step. This is not the case of flow past a static cylinder, where the domain mesh is removed as per the shape and location of the structure. The mesh used in this study is done by creating the mesh of fluid domain and structure separately and then defining the intial location of the structure in the fluid domain by snapping its mesh into the mesh of the fluid domain. This can be done using the *snappyHexMesh* dictionary of OpenFOAM. The fluid domain mesh is created using the blockMesh dictionary and the structure mesh (here cylinder) is created using the open-source meshing

software, Salome v9.10.0. The entire process of this type of meshing in explained in the following section.

snappyHexMesh

This type of 3D meshing is used for complex geometry where the domain mesh and mesh of the complex structure is constructed separately and the structure is snapped in to the domain mesh. It generates 3D meshes containing hexahedra and split hexahedra from a dictionary file named snappyHexMeshDict located in the system directory and a triangulated surface geometry file in Stereolithography (STL) format located in the directory constant/triSurface. The steps to create a 3D mesh using *snappyHexMesh* are as follows:

a. Background or domain mesh generation: This is created using blockMesh. The mesh must consist hexes only. The aspect ratio of cell should be approximately 1 near the STL surface at least. At least one intersection of a cell edge should be there with the STL surface.
b. Geometry definition: The STL geometry can be constructed from any geometry modeling tool and can be made up of a single surface or multiple surfaces, using local refinement in each individual surface that describe the geometry. The STL geometry is present in the directory constant/triSurface
c. Castellated mesh or cartesian mesh generation (castellatedMesh)
d. Snapped mesh or body fitted mesh generation (snap)
e. Addition of layers close to the surfaces (addLayers)
f. Checking mesh quality

The meshing utility *snappyHexMesh* reads the dictionary snappyHexMeshDict located in the system directory. The castellation, snapping, and boundary layer meshing steps (c,d,e) are controlled by this dictionary. The final mesh, that is, after boundary layer meshing, should always be located in the directory constant/polyMesh. There are more than 70 parameters that can be controlled in snappyHexMeshDict dictionary, completely in a Text User Interface (TUI).

The refinement level (volume and surface refinement) of the structure is done in reference to the background or domain mesh. The structure used in this work is a circular cylinder with an aspect ratio 2. The geometry and meshing of the structure is shown in Fig. 5.3.2.2. Fig. 5.3.2.1 shows the cross sectional geometry of the structure with 72 nodes on the circumference and the Fig. 5.3.2.2 shows the entire geometry of the structure. The entire geometry consists of 7484 cells.

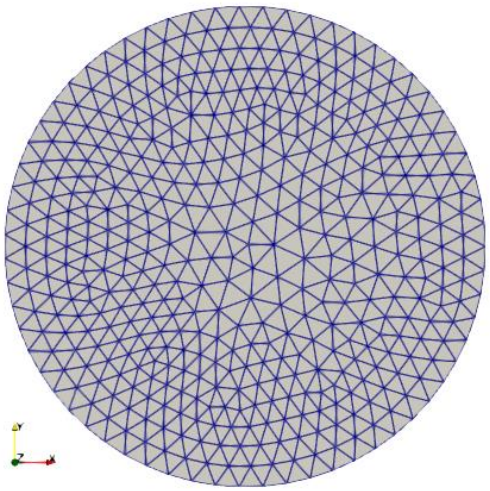


Fig. 5.3.2.1: Cross sectional geometry of the structure

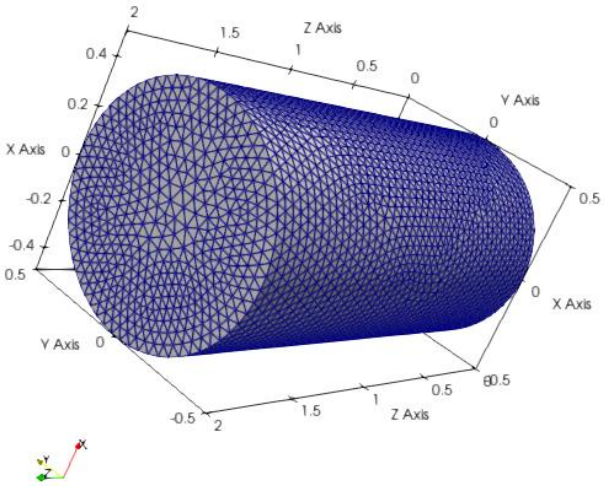


Fig. 5.3.2.2: Entire geometry of the structure

Step 1: Creating the hexahedral background mesh

The present work is an application of external aerodynamics, so an external mesh is created using snappyHexMeshDict. The objective is to mesh a cuboid shaped region or domain shown in Fig. 5.3.2.3 below, surrounding an object or structure described by a STL surface, shown in Fig. 5.3.2.2. Before *snappyHexMesh* is executed, a background mesh of hexahedral cells filling the entire region should be created as shown in the Fig. 5.3.2.3 and the criteria as mentioned earlier should be maintained.

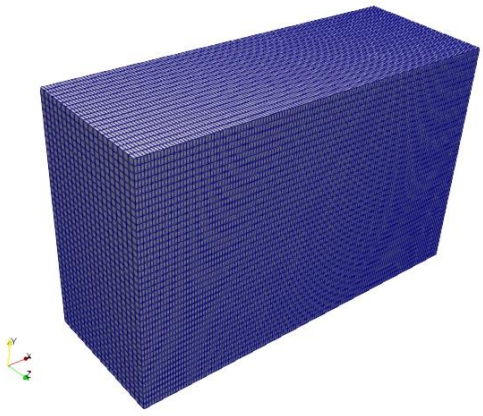


Fig. 5.3.2.3: Cuboid shaped domain mesh

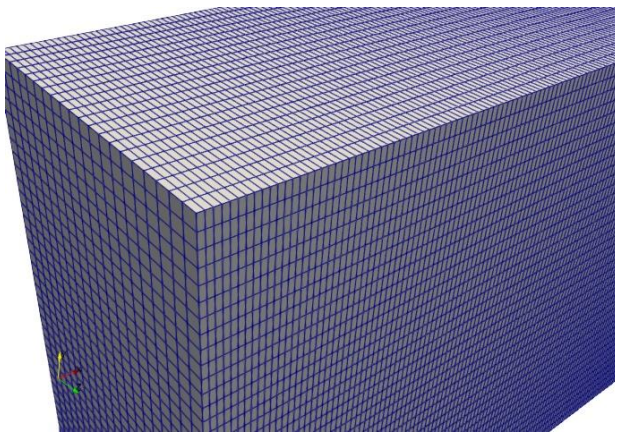


Fig. 5.3.2.4: Domain mesh zoomed in

If the fluid domain mesh and cylinder mesh are represented separately, it looks like Fig. 5.3.2.5.

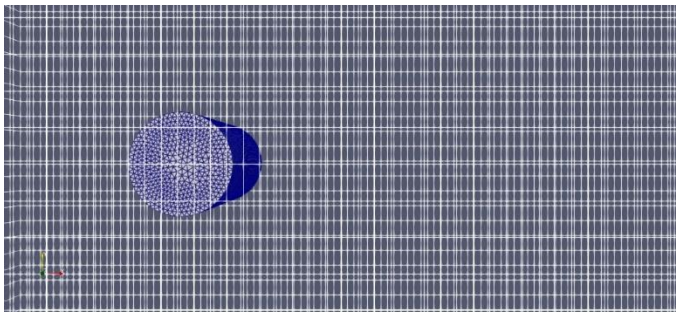


Fig. 5.3.2.5: Structure and background mesh in not castellated or not snapped manner

From the Fig. 5.3.2.5, it can be seen that the background mesh and structure are not in par. *snappyHexMesh* castellates and snaps the structure into the background domain mesh by adding boundary layers. The *snappyHexMesh* dictionary used in the present work are shown in fragments from Fig. 5.3.2.6 onwards.

```
// Which of the steps to run
castellatedMesh true;
snap            true;
addLayers       true;

//Optional: single region surfaces get patch names according to
//          surface only. Multi-region surfaces get patch name
//          surface "_ "region. Default is true
//singleRegionName false;

// Geometry. Definition of all surfaces. All surfaces are of class
// searchableSurface.
// Surfaces are used
// - to specify refinement for any mesh cell intersecting it
// - to specify refinement for any mesh cell inside/outside/near
// - to 'snap' the mesh boundary to the surface
geometry
{
    cylinder.stl
    {
        type triSurfaceMesh;
        name cylinder;

        //tolerance   1E-5;   // optional:non-default tolerance on intersections
        //maxTreeDepth 10;    // optional:depth of octree. Decrease only in case
                              // of memory limitations.

        // Per region the patchname. If not provided will be <surface>_<region>.
        // Note: this name cannot be used to identity this region in any
        //       other part of this dictionary; it is only a name
        //       for the combination of surface+region (which is only used
        //       when creating patches)
        regions
        {
```

Fig. 5.3.2.6: Initiation of snappyHexMeshDict, which steps to run and the extraction of the structure geometry in .stl format present in /constant/triSurface

```
        regions
        {
/*
            secondSolid                 // Named region in the STL file
            {
                name mySecondPatch;     // User-defined patch name
            }
*/
        }
    }

    refinementBox
    {
        type searchableBox;
        min  (-3.0 -2 -2);
        max  ( 8.0  2 2.5);
    }

/*
    refinementBox1
    {
        type searchableBox;
        min (-2 -2 -2);
        max (2 2 2);
    }
*/

};
```

Fig. 5.3.2.7: The refinement box is the region of the background mesh that gets refined to stay in par to the mesh of the structure or body.

Step-2: Castellation

Step 2a: Cell splitting at feature edges: The snappyHexMeshDict dictionary's castellatedMeshControls sub-dictionary allows the user to specify how cells should be fragmented. Cells are first chosen for splitting based on specified edge properties. The surfaceFeatureExtract utility is used to extract feature edges from the STL geometry file.

```
// Settings for the castellatedMesh generation.
castellatedMeshControls
{

    // Refinement parameters
    // ~~~~~~~~~~~~~~~~~~~~~

    // If local number of cells is >= maxLocalCells on any processor
    // switches from from refinement followed by balancing
    // (current method) to (weighted) balancing before refinement.
    maxLocalCells 100000;

    // Overall cell limit (approximately). Refinement will stop immediately
    // upon reaching this number so a refinement level might not complete.
    // Note that this is the number of cells before removing the part which
    // is not 'visible' from the keepPoint. The final number of cells might
    // actually be a lot less.
    maxGlobalCells 2000000;

    // The surface refinement loop might spend lots of iterations refining just a
    // few cells. This setting will cause refinement to stop if <= minimumRefine
    // are selected for refinement. Note: it will at least do one iteration
    // (unless the number of cells to refine is 0)
    minRefinementCells 0;

    // Allow a certain level of imbalance during refining
    // (since balancing is quite expensive)
    // Expressed as fraction of perfect balance (= overall number of cells /
    // nProcs). 0=balance always.
    maxLoadUnbalance 0.10;

    // Number of buffer layers between different levels.
    // 1 means normal 2:1 refinement restriction, larger means slower
    // refinement.
    //nCellsBetweenLevels 1;
    nCellsBetweenLevels 2;
```

```
    // Explicit feature edge refinement
    // ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

    // Specifies a level for any cell intersected by explicitly provided
    // edges.
    // This is a featureEdgeMesh, read from constant/triSurface for now.
    // Specify 'levels' in the same way as the 'distance' mode in the
    // refinementRegions (see below). The old specification
    //      level   2;
    // is equivalent to
    //      levels  ((0 2));

    features
    (
        {
            file "cylinder.eMesh";      //cylinderwithbase.eMesh
            level 4;
          //level 4;
        }
    );
```

Fig. 5.3.2.8: Castellated mesh generation

Step 2b: Cell splitting at surfaces: After feature edge refinement, cells are chosen for splitting in the locality of specified surfaces. The surface refinement (splitting) is specified in the refinementMeshControls in the castellatedMeshControls sub-dictionary in the snappyHexMeshDict dictionary.

```
// Surface based refinement
// ~~~~~~~~~~~~~~~~~~~~~~~~

// Specifies two levels for every surface. The first is the minimum level,
// every cell intersecting a surface gets refined up to the minimum level.
// The second level is the maximum level. Cells that 'see' multiple
// intersections where the intersections make an
// angle > resolveFeatureAngle get refined up to the maximum level.

refinementSurfaces
{
    cylinder
    {
        // global
        // Surface-wise min and max refinement level
        level (2 2);
    }
        //- Optional increment (on top of max level) in small gaps
        //gapLevelIncrement 2;

        //- Optional angle to detect small-large cell situation
        //  perpendicular to the surface. Is the angle of face w.r.t.
        //  the local surface normal. Use on flat(ish) surfaces only.
        //  Otherwise leave out or set to negative number.
        //perpendicularAngle 10;

        //- Optional faceZone and (for closed surface) cellZone with
        //  how to select the cells that are in the cellZone
        //  (inside / outside / specified insidePoint)
        //  The orientation of the faceZone is
        //  - if on cellZone(s) : point out of (maximum) cellZone
        //  - if freestanding   : oriented according to surface

        //faceZone sphere;
        //cellZone sphere;
        //mode inside;  //outside/insidePoint
```

```
        //faceZone sphere;
        //cellZone sphere;
        //mode inside;  //outside/insidePoint

        //- Optional specification of what to do with faceZone faces:
        //      internal : keep them as internal faces (default)
        //      baffle   : create baffles from them. This gives more
        //                 freedom in mesh motion
        //      boundary : create free-standing boundary faces (baffles
        //                 but without the shared points)
        //faceType baffle;
}
// Feature angle:
// - used if min and max refinement level of a surface differ
// - used if feature snapping (see snapControls below) is used
//resolveFeatureAngle 10;
//resolveFeatureAngle 30;           //default
  resolveFeatureAngle 60;           //To avoid too much refinement with curvature
//resolveFeatureAngle 90;

//- Optional increment (on top of max level) in small gaps
//gapLevelIncrement 2;

// Planar angle:
// - used to determine if surface normals
//   are roughly the same or opposite. Used
//      - in proximity refinement
//      - to decide when to merge free-standing baffles
//        (if e.g. running in surfaceSimplify mode set this to 180 to
//         merge all baffles)
//      - in snapping to avoid snapping to nearest on 'wrong' side
//        of thin gap
//
// If not specified same as resolveFeatureAngle
planarAngle 30;
```

Fig. 5.3.2.9: Surface based refinement

Step 2c: Cell removal: After completion of Step 2a and 2b, a process of cell removal begins. The locationInMesh keyword in the castellatedMeshControls sub-dictionary in the

snappyHexMeshDict dictionary, as shown in Fig. 5.3.2.10, identifies the region in which cells are retained. Cells are retained if around 50% or more of their volume lies within the region.

```
    // Mesh selection
    // ~~~~~~~~~~~~~~~~

    // After refinement patches get added for all refinementSurfaces and
    // all cells intersecting the surfaces get put into these patches. The
    // section reachable from the locationInMesh is kept.
    // NOTE: This point should never be on a face, always inside a cell, even
    // after refinement.
    locationInMesh (5.0 0 1);

    // Whether any faceZones (as specified in the refinementSurfaces)
    // are only on the boundary of corresponding cellZones or also allow
    // free-standing zone faces. Not used if there are no faceZones.
    allowFreeStandingZoneFaces true;
    //allowFreeStandingZoneFaces false;

    // Optional: do not remove cells likely to give snapping problems
    // handleSnapProblems false;

    // Optional: switch off topological test for cells to-be-squashed
    //           and use geometric test instead
    //useTopologicalSnapDetection false;
}
```

Fig. 5.3.2.10: locationInMesh in the castellatedMeshControls sub-dictionary in the snappyHexMeshDict dictionary

Step 2d: Cell splitting in specified regions: Other than the refinement box, the cells are further split into smaller cells near the wall of the structure. The information related to the refinement of the volume regions can be given as input in the refinementRegions block in the castellatedMeshControls sub-dictionary in the snappyHexMeshDict dictionary as shown in Fig. 5.3.2.11.

```
// Region-wise refinement
// ~~~~~~~~~~~~~~~~~~~~~~~~~~

// Specifies refinement level for cells in relation to a surface. One of
// three modes
// - distance. 'levels' specifies per distance to the surface the
//   wanted refinement level. The distances need to be specified in
//   increasing order.
// - inside. 'levels' is only one entry and only the level is used. All
//   cells inside the surface get refined up to the level. The surface
//   needs to be closed for this to be possible.
// - outside. Same but cells outside.

refinementRegions
{
    refinementBox
    {
        mode inside;                // inside - outside
        levels ((1 1));             // min max

        //mode distance;            //
        //levels ((0.1 1));         //first number distance normal to the surface in both direction
    }
}
```

Fig. 5.3.2.11: refinementRegions block in the castellatedMeshControls sub-dictionary in the snappyHexMeshDict dictionary

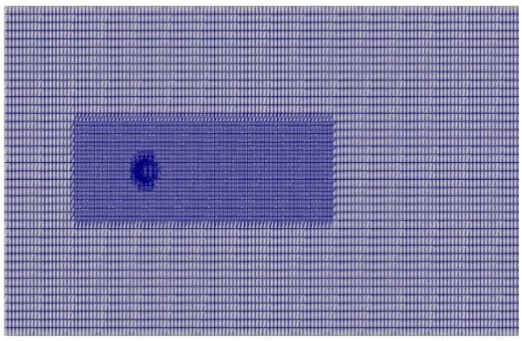


Fig. 5.3.2.12: Refinement Box shown in the entire domain

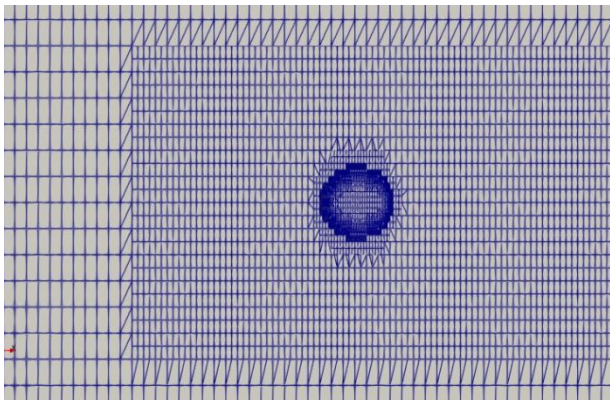


Fig. 5.3.2.13: Refinement Box (zoomed in)

These steps (2a, 2b, 2c, 2d) form a valid castellated or cartesian mesh that can be used for a simulation. The castellated mesh is shown in Fig. 5.3.2.14.

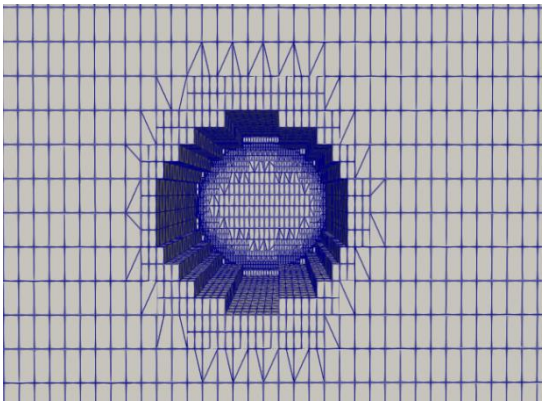


Fig. 5.3.2.14: Castellated mesh

Step-3: Snapping

A conforming mesh is created by snapping the points on the surface after steps 2c and 2d. This is given as input in the snapControls sub-dictionary in snappyHexMeshDict, as shown in Fig. 5.3.2.15.

```
// Settings for the snapping.
snapControls
{
    // Number of patch smoothing iterations before finding correspondence
    // to surface
    nSmoothPatch 3;     //recommended   //nSmoothPatch 10;      //improved

    // Maximum relative distance for points to be attracted by surface.
    // True distance is this factor times local maximum edge length.
    // Note: changed(corrected) w.r.t 17x! (17x used 2* tolerance)
    tolerance 2.0;
    //tolerance 1.0;

    // Number of mesh displacement relaxation iterations.
    nSolveIter 30;      //recommended
    //nSolveIter 100;   //improved

    // Maximum number of snapping relaxation iterations. Should stop
    // before upon reaching a correct mesh.
    nRelaxIter 5;       //recommended ///nRelaxIter 10; //improved  10-20-50

    // Feature snapping

    // Number of feature edge snapping iterations.
    // Leave out altogether to disable.
    nFeatureSnapIter 10; //recommended //nFeatureSnapIter 100;  //improved  50-100

    // Detect (geometric only) features by sampling the surface
    // (default=false).
    implicitFeatureSnap false;

    // Use castellatedMeshControls::features (default = true)
    explicitFeatureSnap true;

    // Detect features between multiple surfaces
    // (only for explicitFeatureSnap, default = false)
    multiRegionFeatureSnap false;

    // wip: disable snapping to opposite near surfaces (revert to 22x behaviour)
    // detectNearSurfacesSnap false;
}
```

Fig. 5.3.2.15: snapControls sub-dictionary in snappyHexMeshDict

This forms a valid snapped or body fitted mesh as shown in Fig. 5.3.2.16.

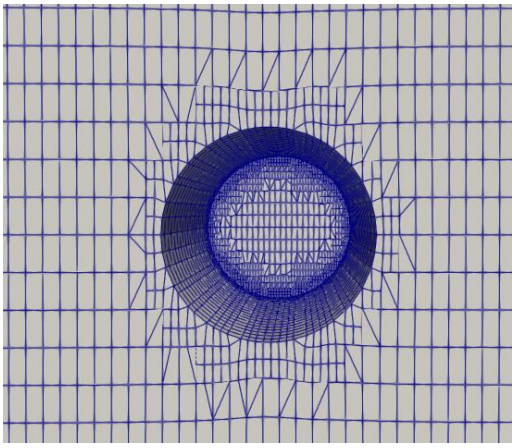


Fig. 5.3.2.16: Snapped or body fitted mesh

Step-4: Mesh layers

Although the mesh created from snapping may be suitable for simulation, but it can create some irregular cells along boundary surfaces. An input for introducing boundary layer meshing in selected parts of the mesh can be given in the addLayersControls sub-dictionary in the snappyHexMeshDict dictionary as shown in Fig. 5.3.2.16 in Annexure-III: Section-5.3.2.

This is the final step of the mesh generation process using *snappyHexMesh*. This is a valid body fitted mesh with boundary layer meshing, that can be used for a simulation. The final mesh is shown in Fig. 5.3.2.17 and Fig. 5.3.2.18.

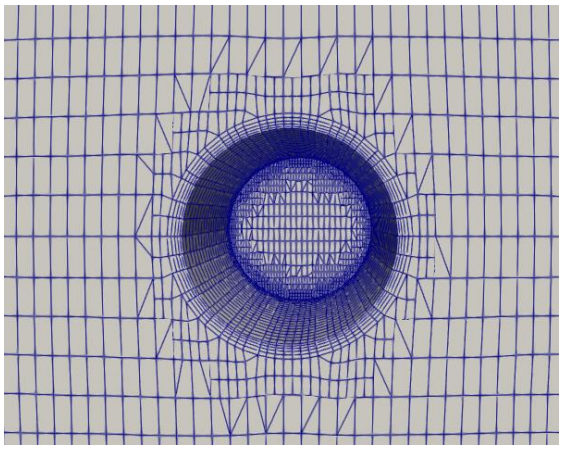


Fig. 5.3.2.17: addLayersControls

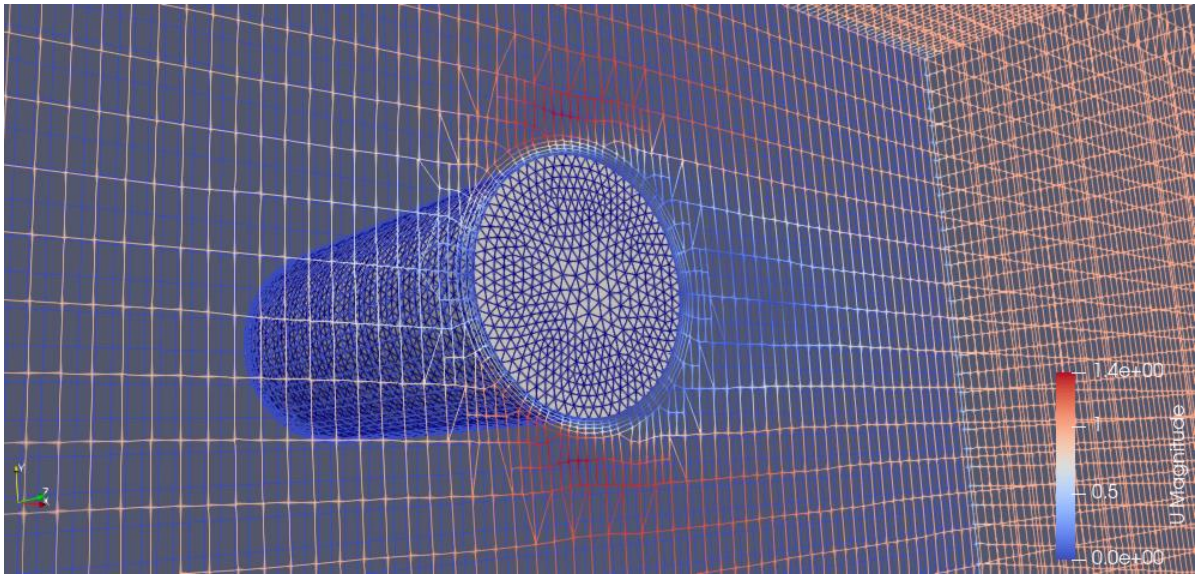


Fig. 5.3.2.18: Final mesh created by *snappyHexMesh* and the original cylinder geometry fitted in it

The final mesh consists of 304745 cells. The simulations for $V_r = 6$ is done for a denser mesh, where the mesh contains 1.8 times more the number of cells, that is 546720 cells. Both the meshes adopt a same structured grid topology with different number of cells. The comparison is done with respect to the amplitude ratio, the root mean square (RMS) lift coefficient (CL_{rms}). The drag coefficient (C_D) and lift coefficients (C_L) are defined in Eq 5.3.1.

$$C_D = \frac{F_x}{\left(\frac{1}{2}\rho A_p U^2\right)} \qquad \text{and, } C_L = \frac{F_y}{\left(\frac{1}{2}\rho A_p U^2\right)} \qquad [5.3.1]$$

F_x and F_y are the drag and lift forces in the in-line and the cross-flow directions, respectively and ρ is the fluid density, A_p is the projected area, U is the free-stream velocity. This comparison is done for $V_r = 6$, the reduced velocity at which the VIV response amplitude is maximum for the cylinder with a finite length in the discussion in the later half of Chapter-5. The numerically computed results from the two meshes agree well with each other. The force coefficients from the two meshes for L/D = 2 are almost identical, which shows that mesh density has small effect and the normal mesh can be used to carry out further studies.

Table 5.3.2: Comparison of response $\left(A_y/D\right)$ and rms lift coefficient $(CL_{rms})$for the normal and dense mesh

Mesh	Cell count	A_y/D	CL_{rms}
Normal	304745	0.760243	0.43002
Dense	546720	0.7470095	0.41903

The comparison of the cylinder amplitude of these two types of mesh are given in Fig. 5.3.2.19.

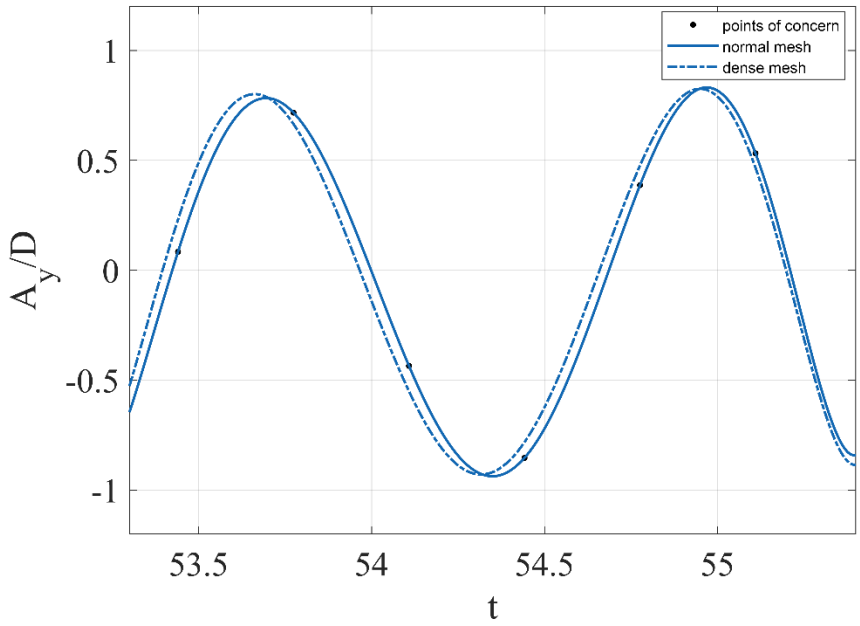


Fig. 5.3.2.19: Comparison of the cylinder amplitude of the two types of mesh (normal and dense); 1DOF

5.3.2.1.3 Time Independence Study

Along with this, the time-step independence study should also be done in order to obtain converged and accurate results. The time steps chosen are 0.01 s and 0.001 s. Every other parameter including the computational mesh are kept the same. The comparison of the time series data in Fig. 5.3.2.20 reveals that the smaller time step 0.001s, should be chosen which is elaborated in Table 5.3.3. The results listed in Table 5.3.3 show that the time-step size of Δt=0.001s is more accurate and is thus adopted in the present study. To keep the Courant number below 1 and preventing the solutions from diverging, 'adjustableTimeStep' option is enabled in system/controlDict of OpenFOAM.

Table 5.3.3: Comparison of response (A_y/D) and rms lift coefficient $(CL_{rms})$for the two time steps, using the normal mesh at $V_r = 6$

Time Step size, s	Cell count	A_y/D, Present	RMS Lift Coefficient, CL_{rms}
0.01	304745	0.790243	0.46508
0.001	304745	0.760243	0.43002

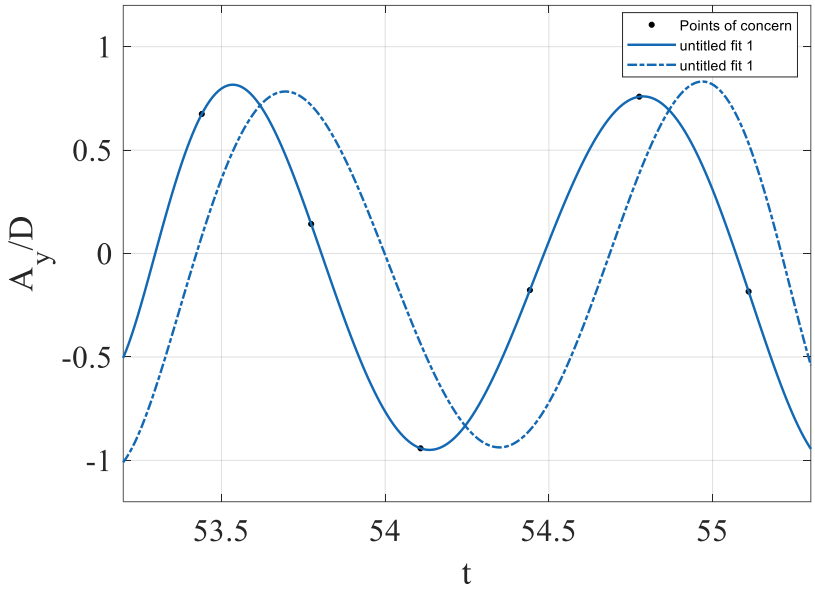


Fig. 5.3.2.20: Comparison of the cylinder amplitude of two time steps chosen; 1DOF

5.3.2.1.4 Validation Tests

The VIV of a circular cylinder with an aspect ratio of 2 at Re = 300 was studied by [20] to analyze the effects of the free end of the cylinder due to the two-way coupling. The present study is validated based on the study done by [20] and [52], without the base column.

To study the intrinsic turbulent characteristics of flow at Re = 300, three-dimensional simulations are performed. This simulation is an example of two-way coupling where the structure (here, circular cylinder) is initially stationary and is subjected to a fluid flow with a particular free-stream velocity. Vortex shedding occurs due to the presence of the body in the fluid and this in turn causes the body to vibrate or oscillate in the unrestrained direction. The vibration of the structure exerts force on the fluid which influence the vortex formation. The mass ratio of the truncated cylinder is set to 2, with zero structural damping. Out of the 6 degrees of freedom of the cylinder, the degree of freedom in the cross-flow direction (y-direction in the axes system opted in this study) of the cylinder is unrestrained with a notional spring of particular stiffness in the system. The flow parameters and intrinsic turbulent characteristics is studied by varying the non-dimensional reduced velocity V_r from 3 to 14 with an increment of 1. Fig. 5.3.2.21 compares the amplitude ratio of the cylinder in cross-flow direction A_y/D, and root mean square lift coefficient CL_{rms} respectively at different V_r from the present simulations with the findings of [20] and validation study of [52]. The results show good agreement with the published results. The maximum response is found at $V_r = 6$.

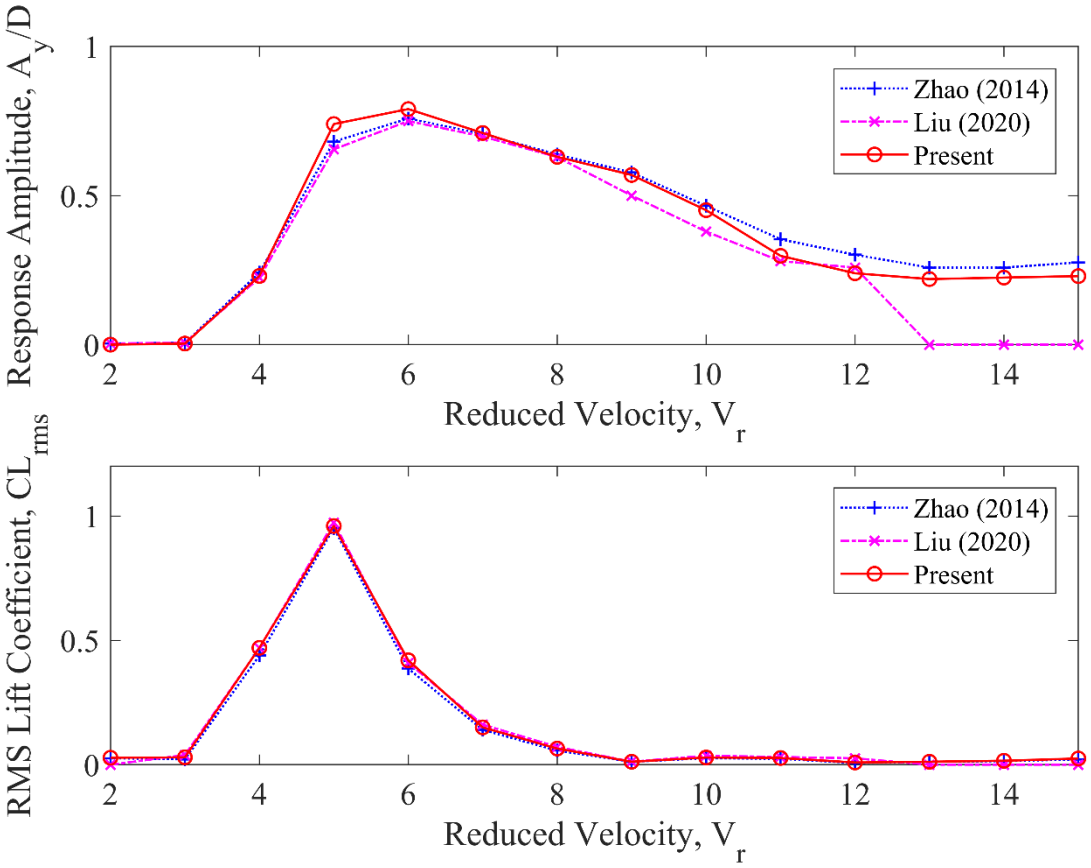


Fig. 5.3.2.21: Validation of the amplitude of the cylinder in cross-flow direction A_y/D, and root mean square lift coefficient CL_{rms} respectively at different V_r of the present simulations with the findings of [20] and validation study of [52]; 1DOF

The Fast Fourier Transform (FFT) of time series for $V_r = 6$ is shown in Fig. 5.3.2.22. The magnitude of rms lift coefficient is in well agreement with the literature followed [20] and [52].

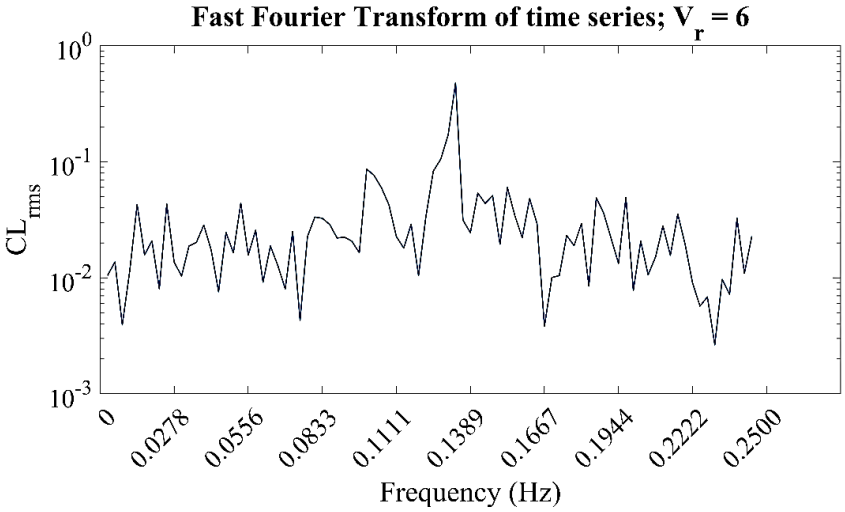


Fig. 5.3.2.22: FFT of time series, CL_{rms}; $V_r = 6$; 1DOF

5.3.2.2 Results and Discussions of VIV of 3D Offshore Spar Platform

The response of the truncated circular cylinder as shown in Fig. 5.3.2.21, is discussed in this section. Three studies are done as mentioned earlier in Chapter-2. For every simulation, the Reynolds number, aspect ratio of cylinder and the mass ratio are fixed at 300, 2 and 2, respectively. The total time up to which the simulations are run is 350 seconds. For each case, the simulation is run for a sufficient amount of time to ensure that at least 25 periods of vibration are achieved for standard vibrations. The usual simulation clock time for each case is 5,50,000 seconds on average.

5.3.2.2.1 First Study:

The variation of amplitude of the cylinder due to the location, arrangement and value of the stiffness of springs, thus, raising the question whether the unrestrainment of the cylinder in the form of putting springs is absolutely notional.

The stiffness of the spring that is used in order to unrestrain the degree of freedom of the cylinder in the cross-flow direction is determined from the reduced velocity chosen. The reduced velocity is given as,

$$\text{Reduced Velocity } (V_r) = \frac{U}{f_n D} \qquad [5.3.2]$$

The free-stream velocity and diameter of the cylinder chosen for this study are 1m/s and 1m. respectively. From the chosen reduced velocity range, the natural frequency (f_n, Hz) of the structure can be calculated for each of the case from Eq 5.3.3.

$$f_n = \frac{U}{V_r D} \qquad [5.3.3]$$

The radial frequency (ω, rad/s) is then calculated, $\omega = 2\pi f_n$. And from that, the stiffness of the system (k, N/m) is evaluated, $k = m\omega^2$

Table-5.3.4: Variation of stiffness as per the reduced velocity

V_r	f_n, Hz	ω, rad/s	k, N/m
3	0.333	2.09	13.8
4	0.25	1.57	7.75
5	0.2	1.26	4.96
6	0.167	1.05	3.45
7	0.143	0.898	2.53
8	0.125	0.785	1.94
9	0.111	0.698	1.53
10	0.1	0.628	1.24
11	0.0909	0.571	1.03
12	0.0833	0.524	0.861
13	0.0769	0.483	0.734
14	0.0714	0.449	0.633

The stiffness of the system depends on the location, arrangement and value of the stiffness of springs. This study is not explicitly available in literature and thus form one of the focus of this study.
The locations of the spring used in this study are: i) At the bottom free-end of the cylinder ii) At the center of the span of the cylinder
The springs can be in series or parallel arrangement, and accordingly the value of spring stiffness change. The number of springs used to reach the desired stiffness of the system depends on the number of springs being used. These informations are necessary, as per the input required in the dynamicMeshDict of OpenFOAM.
The springs are in parallel connection. In the parallel combination, considering two springs, Spring1 and Spring2, the restoring force of the spring are $F_1 = k_1 x_1$ and $F_2 = k_2 x_2$, where k_1, k_2 and x_1, x_2 are the spring stiffness and displacement of each spring. The displacement for each spring is same, so $x_1 = x_2 = x$ and thus, from the equivalent force, the equivalent stiffness can be evaluated;

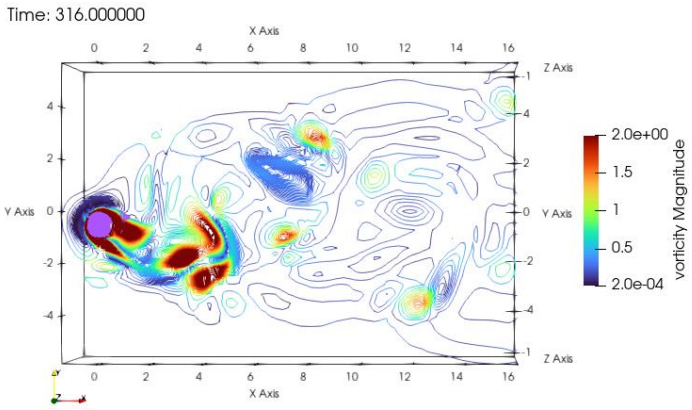
$F(= kx) = F_1 + F_2 = k_1 x + k_2 x$ and, $k = k_1 + k_2$

The first case is of attaching a single spring at the free end of the cylinder. The stiffness of the spring is the total stiffness of the system. The second case is of attaching two springs in parallel connection at mid cross-section of the cylinder and the amplitude ratio of the various cases are shown in Fig. 5.3.2.21.

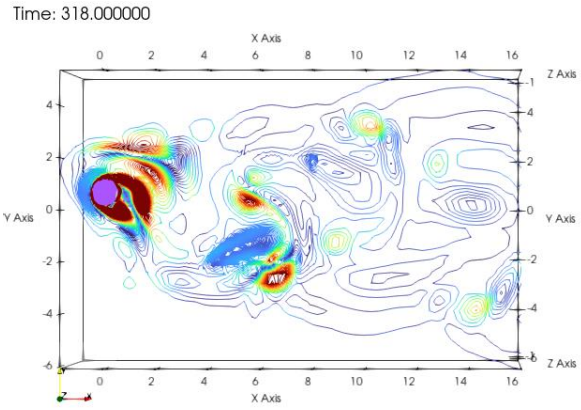
In both these cases, the amplitude ratio, rms lift coefficient give similar results. But, there are slight differences in the nature of oscillation based on the location of the spring. Thus, the stiffness of the system is independent on the number of springs, the stiffness of the system being same and the springs should always be in parallel arrangement. It slightly depend on the location of the spring.

5.3.2.2.2 Second Study:

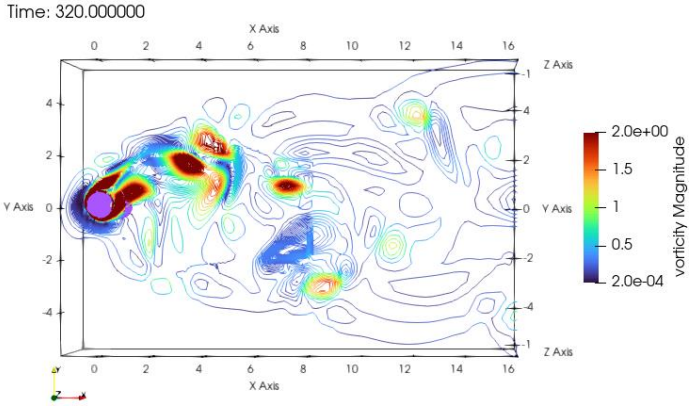
For cylinders of low-aspect ratio, the effects of the free-end of cylinder that leads to the vortex shedding convection towards the top end of the cylinder by the upwash velocity, as mentioned in [20] is also studied in this work.
The contours of the non-dimensional vorticity in the axial direction of the cylinder on the top surface of the computational domain (z = L) are shown in the Fig. 5.3.2.24. Allowing the cylinder to vibrate in the cross-flow direction leads to vortex shedding and vibration at the chosen short cylinder length. The 2D vorticity contours in the axial direction of the cylinder on the top surface of the computational domain is shown in Figs. 5.3.2.24 (a) to (h).


Time: 316.000000
X Axis
Y Axis
Z Axis
2.0e+00
1.5
1
0.5
2.0e-04
vorticity Magnitude

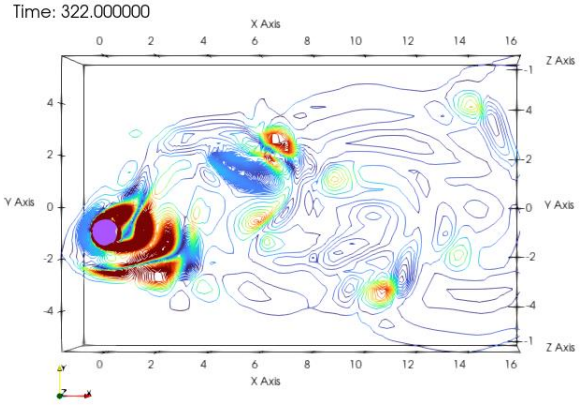
(a)


Time: 318.000000
X Axis
Y Axis
Z Axis

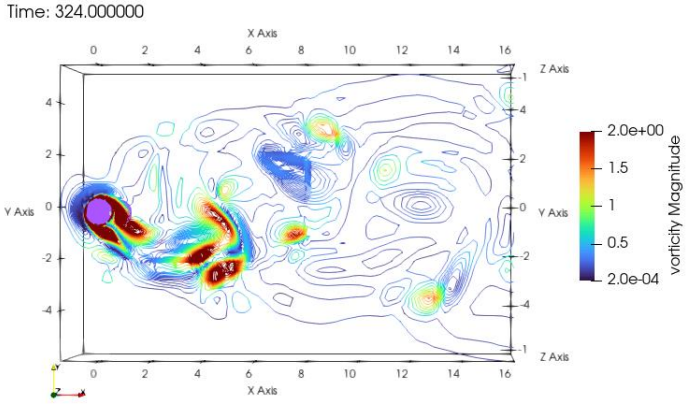
(b)


Time: 320.000000
X Axis
Y Axis
Z Axis
2.0e+00
1.5
1
0.5
2.0e-04
vorticity Magnitude

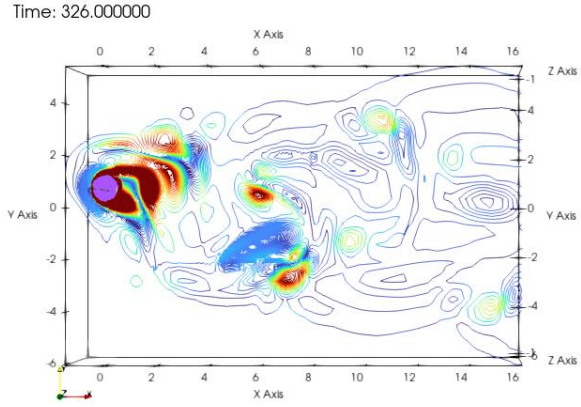
(c)


Time: 322.000000
X Axis
Y Axis
Z Axis

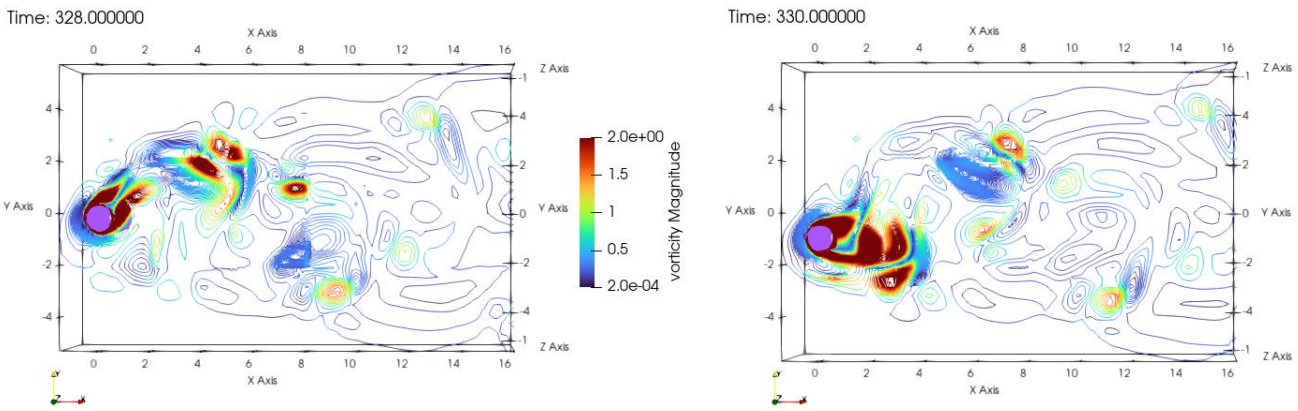
(d)


Time: 324.000000
X Axis
Y Axis
Z Axis
2.0e+00
1.5
1
0.5
2.0e-04
vorticity Magnitude

(e)


Time: 326.000000
X Axis
Y Axis
Z Axis

(f)



(g) (h)

Fig. 5.3.2.23: Contours of non-dimensional axial vorticity around the cylinder at the top surface of the computation domain for L/D=2 and $V_r = 6$, for the time steps shown; 1DOF

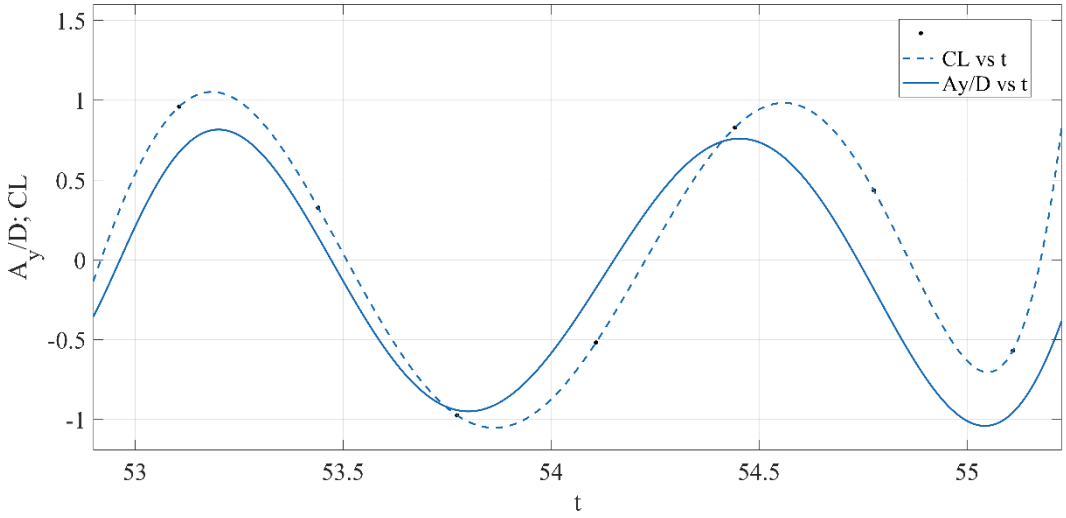


Fig. 5.3.2.24: Time histories of the vibration displacement and the lift coefficient at $V_r = 6$, for the the instants corresponding to the flow structures depicted in Fig. 5.3.2.23 (a to h) ; 1DOF

On the time histories of the vibration displacement and the lift coefficient presented in Fig. 5.3.2.24, the instants corresponding to the flow structures are marked by the dots.

Flow structure around the cylinder is examined to further explain why $V_r < 11$ is classified to be in the lock-in regime. When $V_r = 6$, that is, $V_r < 11$ the vortex shedding frequency and the vibration frequency are same. The lift coefficient and the cylinder displacement are in phase. This is why the vortex shedding in different instances shown in Fig. 5.3.2.24 sustains. The time series of the non-dimensional cylinder response y/D and lift coefficient CL within two periodic response cycles for both the cases is shown in Fig. 5.3.2.25.

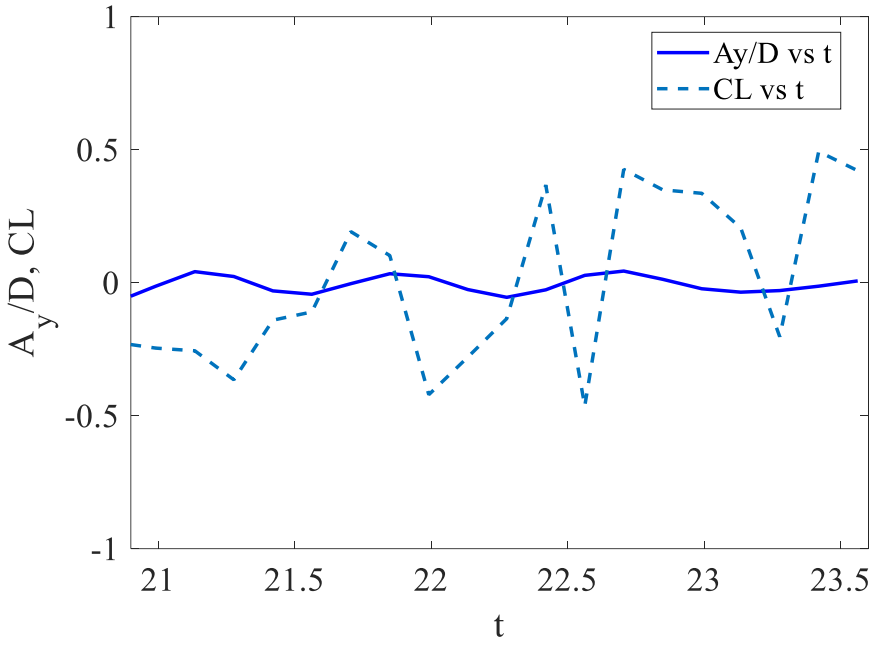


Fig. 5.3.2.25: Time histories time series of the non-dimensional cylinder response y/D and lift coefficient CL within two periodic response cycles at $V_r = 14$; 1DOF

But, when $Vr \geq 11$, it cannot be classified to be in the lock-in regime. The time histories shown in Fig. 5.3.2.25, show that the vortex shedding frequency and the vibration frequency are the same between initial instances, and the lift coefficient is anti-phase with the cylinder displacement. The anti-phase between the lift coefficient and the vibration frequency, on the other hand, does not last. The frequency of the lift coefficient is nearly twice the vibration frequency for the period in the latter instances showing that no lock-in occurs.

The identification and location of a vortex in a flow field is difficult to undestand or detect, when the Q-criterion is required. In several situations, including a turbulent wake, propeller wake, wing downwash, free end of floating offshore structures, visualisation of the coherent vortical structures in the flow can help to comprehend the flowfield. Often, the vorticity magnitude is utilised to try to visualise these structures using a maximum vorticity threshold or vorticity isosurfaces.But there are drawbacks to these, in addition to not so fulfilling coherent structures. The most used method is the Q-criterion. In tensor notation, the value *Q* is obtained from the definition of the velocity gradient tensor δui / δxj which can be decomposed into two parts as:

$$\frac{\partial u_i}{\partial x_j} = \frac{1}{2}\left[\frac{\partial u_i}{\partial x_j} + \frac{\partial u_i}{\partial x_j}\right] + \frac{1}{2}\left[\frac{\partial u_i}{\partial x_j} - \frac{\partial u_i}{\partial x_j}\right] \qquad [5.1]$$

where the symmetric part is known as the strain rate tensor, denoted as *S and* defined by

$$S = ½ \left[\frac{\partial u_i}{\partial x_j} + \frac{\partial u_i}{\partial x_j}\right] \qquad [5.2]$$

and the antisymmetric part known as the rotation rate or vorticity tensor, denoted as Ω and defined by

$$\Omega = \frac{1}{2}\left[\frac{\partial u_i}{\partial x_j} - \frac{\partial u_i}{\partial x_j}\right] \quad [5.3]$$

From the definition of the viscous stress tensor,

$$\tau = \mu\left[\frac{\partial u_i}{\partial x_j} + \frac{\partial u_i}{\partial x_j}\right] \quad [5.4]$$

which are functions of the strain rate only.
From this perspective, Q can be defined as the second invariant of the velocity gradient tensor

$$Q = \frac{1}{2}(\|\Omega\|^2 - \|S\|^2) \quad [5.5]$$

The positive values of Q indicate areas in the flowfield where the vorticity dominates and the negative values of Q indicate strain rate or viscous stress dominated areas.

The three-dimensional vortex flow structures at $V_r = 6$, corresponding to the instant when the cylinder is at its lowest position are presented by the iso-surfaces of the non-dimensional eigenvalue of −0.1 in Fig. 5.3.2.26. The original position of the cylinder is shown as a solid cylinder and the new position of the cylinder is shown alongside. The top view of the instant is shown in Fig. 5.3.2.26.1. The change in position from the original position of the cylinder is shown in Fig. 5.3.2.26.2. The magnified image is shown in Fig. 5.3.2.26.3, where the vortex shedding is clearly understood. The vortices that are almost perpendicular to the cylinder span dominate the wake flow. The direction of the vortices bend towards the axial direction of the cylinder at the top end level of the cylinder.

Fig. 5.3.2.26.1: Top view of the instant

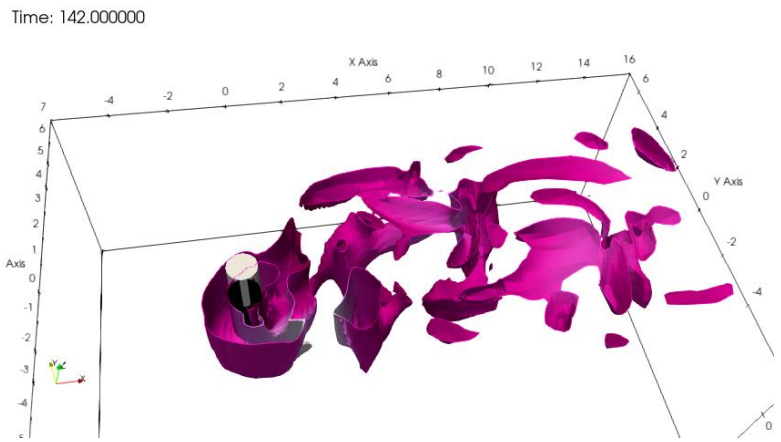


Fig. 5.3.2.26.2: Change in position from the original position of the cylinder

Fig. 5.2.2.26.3: Different view of image

Fig. 5.3.2.26: Isosurfaces of the nondimensional eigenvalue -0.1 when the cylinder is at its lowest position in the cross-flow direction at $V_r = 6$; 1DOF

End Effects: The vibrations of a finite cylinder in a fluid flow with different aspect ratios have been in studied in [20]. The vortex shedding flow from the free-end found in this study is similar to that of [20], but the objective of both the studies are not exactly the same. As the cylinder chosen here has a small length, so the end effects on the vibration are more prominent. From Fig. 5.3.2.27, it can be seen that the vortices in the wake of the cylinder are generated from the free end of the cylinder. These vortices generated from the free end leads to more vortex-induced vibrations. To address this, [20] carried out similar simulations at different aspect ratio of cylinder and studied the end effects.

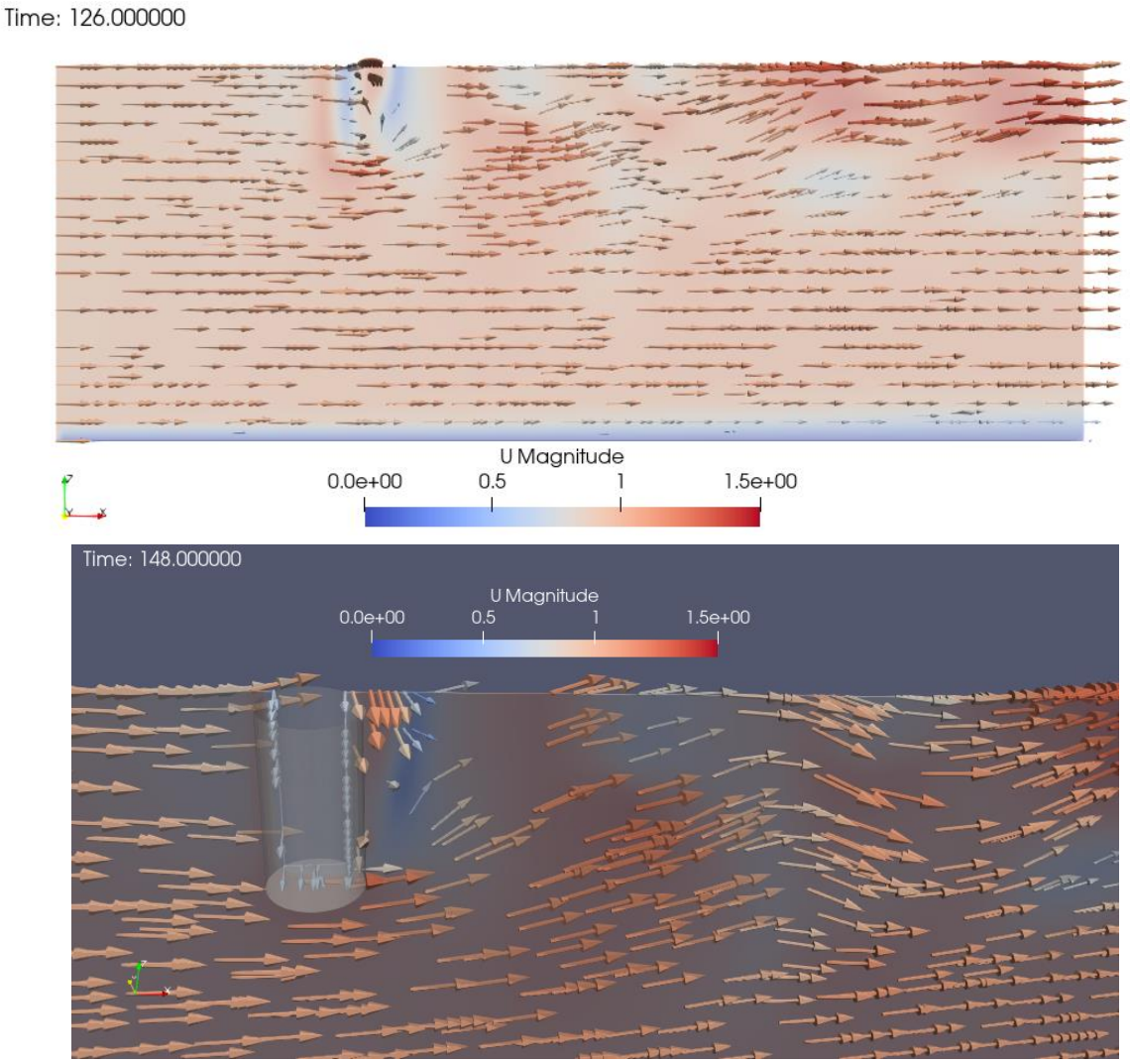


Fig. 5.3.2.27: Vortices in the wake of the cylinder are generated from the free end of the cylinder; 1DOF

The three-dimensional flow structure in Fig. 5.3.2.28 reveals periodic vortex shedding. In Fig. 5.3.2.28 (a to h), the iso-surfaces of the non-dimensional eigenvalue of $e_2 = 0.1$ at 8 instants are visualized from the negative z-direction so as to observe the formation of the vortices near the free end of the cylinder. The instants corresponding to the vortex flows are indicated by the dots on the time histories of the vibration displacement (solid line) and the lift coefficient (dotted line) in Fig. 5.3.2.29. As the cylinder is considered to be a floating structure, the free-end is at the bottom, the vortices from the end of the cylinder lead to an upward velocity field behind the cylinder. This upward velocity field is called upwash, as used by [20]. This is why the vortex shedding is shown for the inverted cylinder, with the negative z-axis pointing upwards.

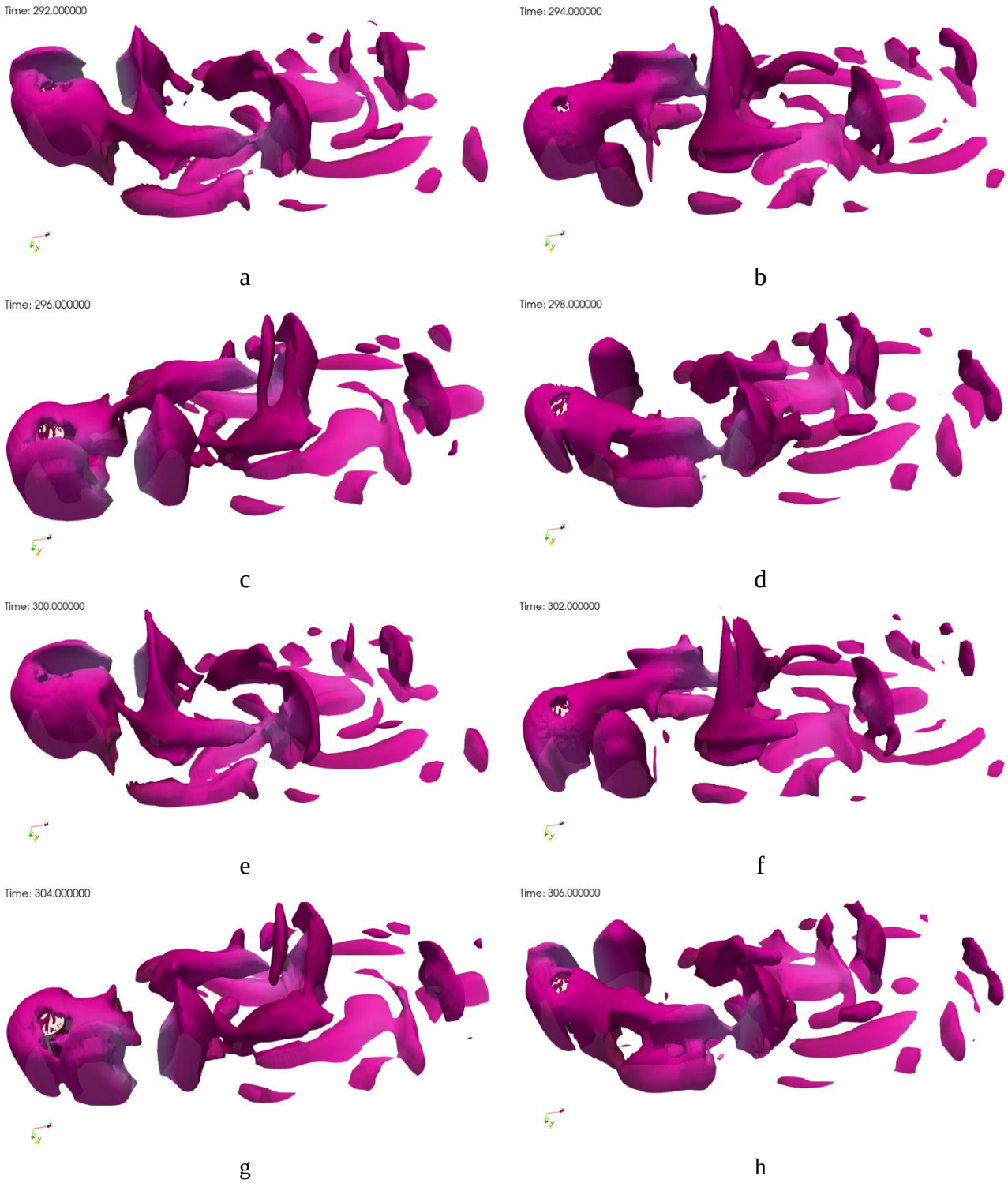


Fig. 5.3.2.28: Iso-surfaces of the non-dimensional eigenvalue of $e_2 = 0.1$ within half period of vibration for $V_r = 6$ visualized from the negative z-direction; 1DOF

This upwash has a considerable impact on the vibration of a free-end cylinder. The vortex formed from the cylinder base as can be seen in Fig. 5.3.2.28 (a), extends towards the positive z-direction

and merges with the vortex formed by the cylinder's negative y-side due to upwash. The energy of the Vortex in the x-y plane (Fig. 5.3.2.28 (a)) feeds into the vortex parallel to the cylinder, enhancing vortex shedding, as seen in Fig. 5.3.2.28 (b) and Fig. 5.3.2.28 (c). It separates from the cylinder at the next instance (c) in Fig. 5.3.2.28 and another Vortex-2 is generated at the positive x-side of the cylinder. This new Vortex-2 repeats what Vortex-1 had done in the previous half period of vibration and does the same in the next half period of vibration (c and d). It can be seen in Figs. 5.3.2.26 (c) and 5.3.2.28 (e) that the vortices in the wake of the cylinder are almost perpendicular to the cylinder. From Fig-5.3.2.28 (a), it can be seen that when a vortex is created, it is diagonally aligned. The downstream component of the vortex is washed to the positive z-direction by the upwash, resulting in a vortex that is practically perpendicular to the cylinder. Asymmetric and dynamic vortex structure is seen in the wake of the vibrating cylinder. The lift force oscillates as a result of the dynamic vortex flow. When a vortex is developed in the positive y-axis, positive lift force is generated and negative lift force is generated when the vortex is developed from the negative y-axis.

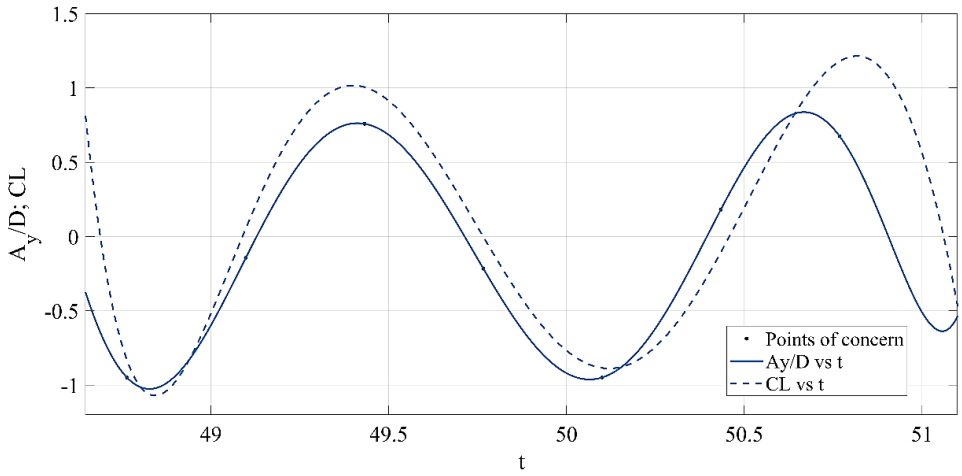


Fig. 5.3.2.29: Time histories of the vibration displacement and the lift coefficient at $V_r = 6$, for the the instants corresponding to the flow structures depicted in Fig. 5.3.2.28 (a to h) ; 1DOF

5.3.2.2.3 Third Study:

The cylinder is unrestrained in the inline direction along with the cross-flow direction. Thus 2DOF VIV of cylinder is considered here. The amplitude ratio, rms lift coefficient and FFT is obtained at the reduced velocity where the maximum amplitude is obtained for the case of 5.3.2.2.2. The comparison of the mesh morphing at the same time step (316 s.) for the 1DOF and 2DOF cases are shown in Fig. 5.3.2.30 and Fig. 5.3.2.31 respectively. The initial position of the cylinder (time = 0 s.) is shown as the solid blue figure in Fig. 5.2.2.30 and Fig. 5.2.2.31.

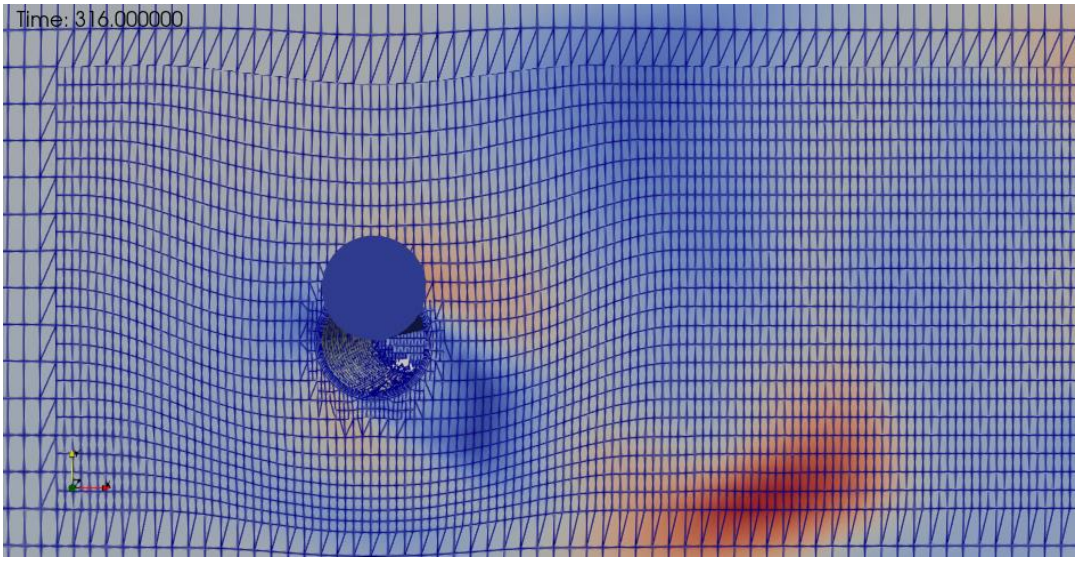


Fig. 5.2.2.30: Morphed mesh at time 316 s. for single DOF system; 1DOF

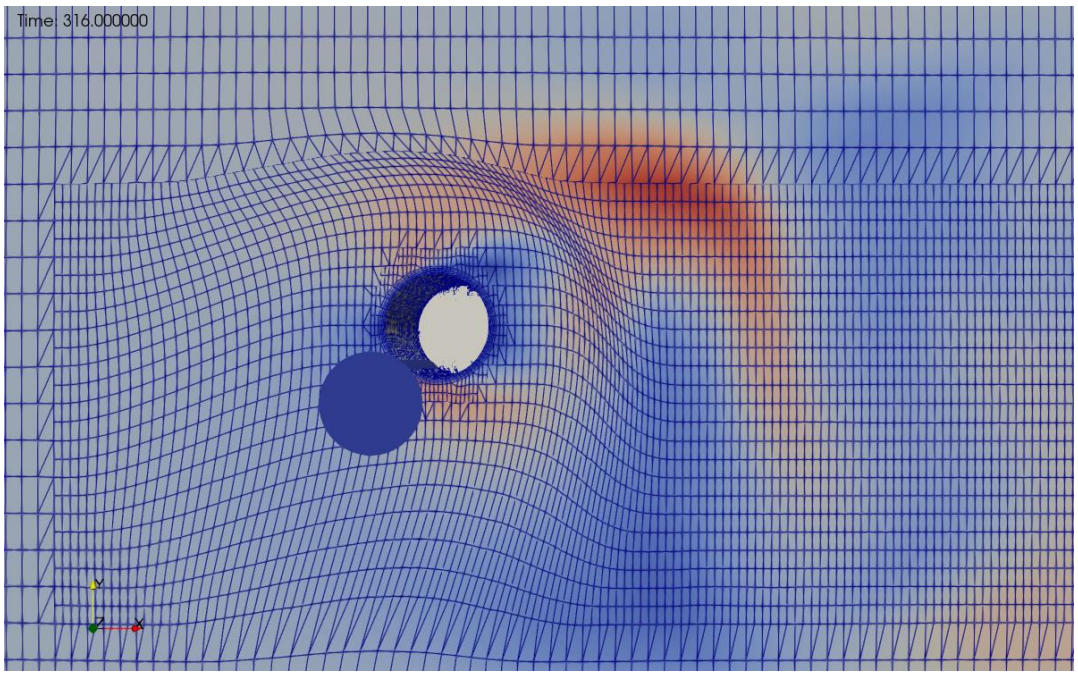


Fig. 5.2.2.31: Morphed mesh at time 316 s. for multi DOF system; 2DOF

The mesh is morphing only in cross-flow direction (y-axis) for the study of Section-5.3.2.2.2 as can be seen in Fig. 5.3.2.30. But, for the study of Section-5.3.2.2.3 the mesh is morphing in both in-line (x-axis) and cross-flow direction (y-axis) as can be seen in Fig. 5.3.2.31, thus it is referred as '2DOF'.

The time-history for the cylinder response in the in-line direction and in the cross-flow direction are depicted in Fig. 5.3.2.32. As the flow is from the inlet along the x-direction, so the response in the in-line direction initially is maximum and then reduces. Again, the response in the cross-flow direction have less amplitude initially and then reaches a regularity, for the same reason. The

amplitude of the cylinder in inline direction is found to be 0.56D and in the cross-flow direction to be 0.78D, which is near to that of the 1DOF case.

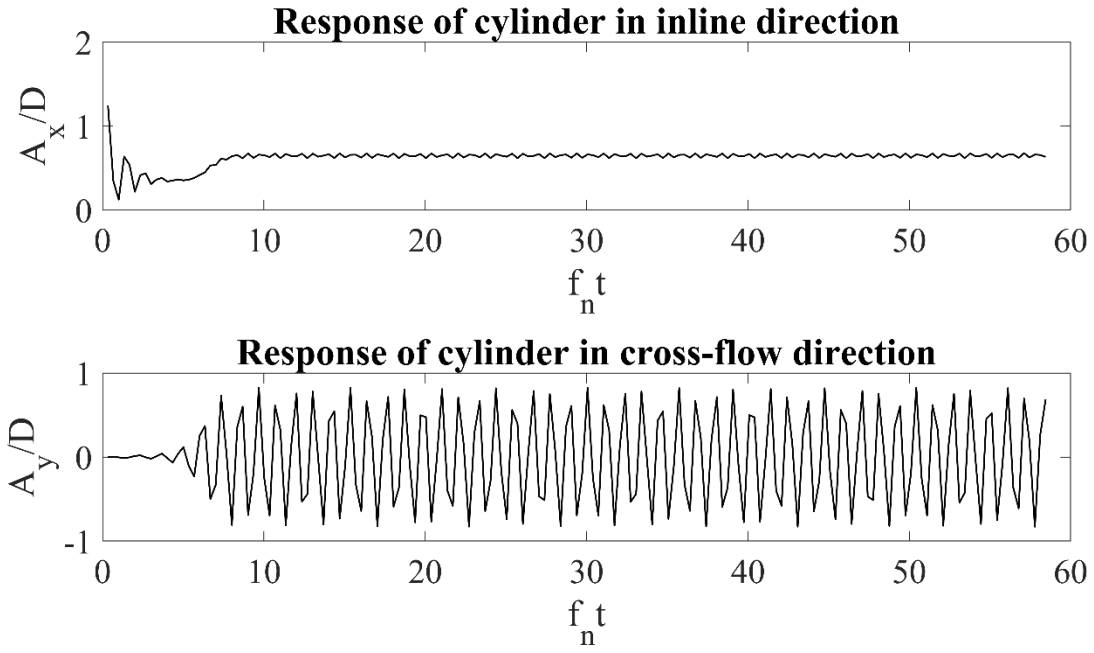
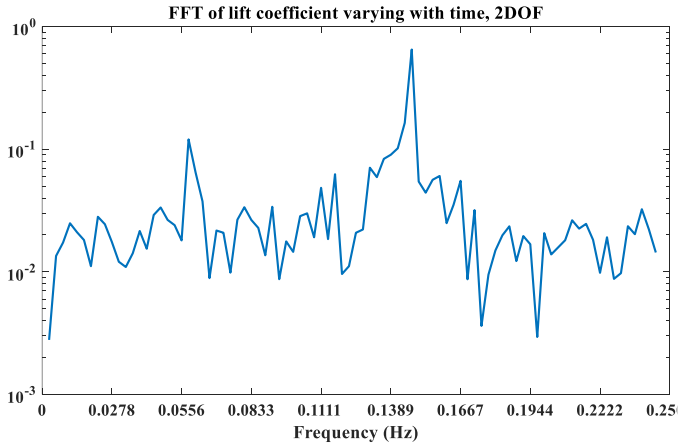


Fig. 5.3.2.32. Time-history for the cylinder response in the in-line direction (A_x) and in the cross-flow direction (A_y); f_n represents the natural frequency of the structure and t represents the time in seconds; 2DOF

The Fast Fourier Transform (FFT) of time series, RMS lift coefficient (CL_{rms}) and RMS drag coefficient (CD_{rms}) for $V_r = 6$ is shown in Fig. 5.3.2.33.1 and Fig. 5.3.2.33.2. In Fig. 5.3.2.33.2, the magnitude of CD_{rms} is highest at the beginning, because the fluid flow, parallel to the x-direction, at the inlet starts at the onset of the simulation has started. So, the fluid at first exerts a horizontal force to the cylinder at rest, which leads to displacement of cylinder in the inline directio first and then the cylinder starts oscillating.



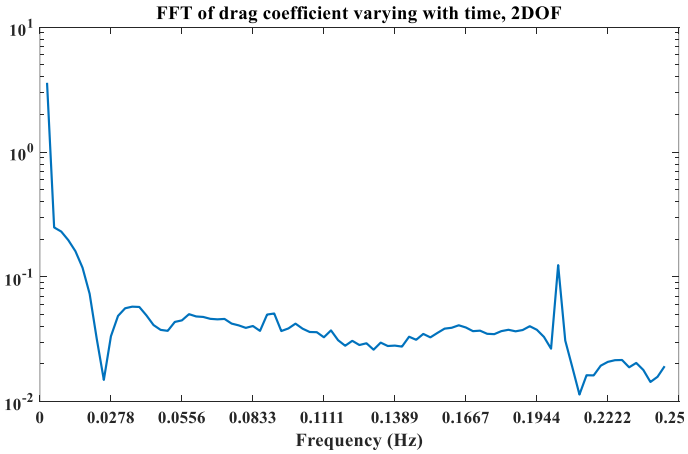


Fig. 5.3.2.33.1: FFT of time series, CL_{rms}; $V_r = 6$; 2DOF

Fig. 5.3.2.33.2: FFT of time series, CD_{rms}; $V_r = 6$; 2DOF

A large recirculation zone can be observed from the free end to wake of the cylinder in Fig. 5.3.2.34.

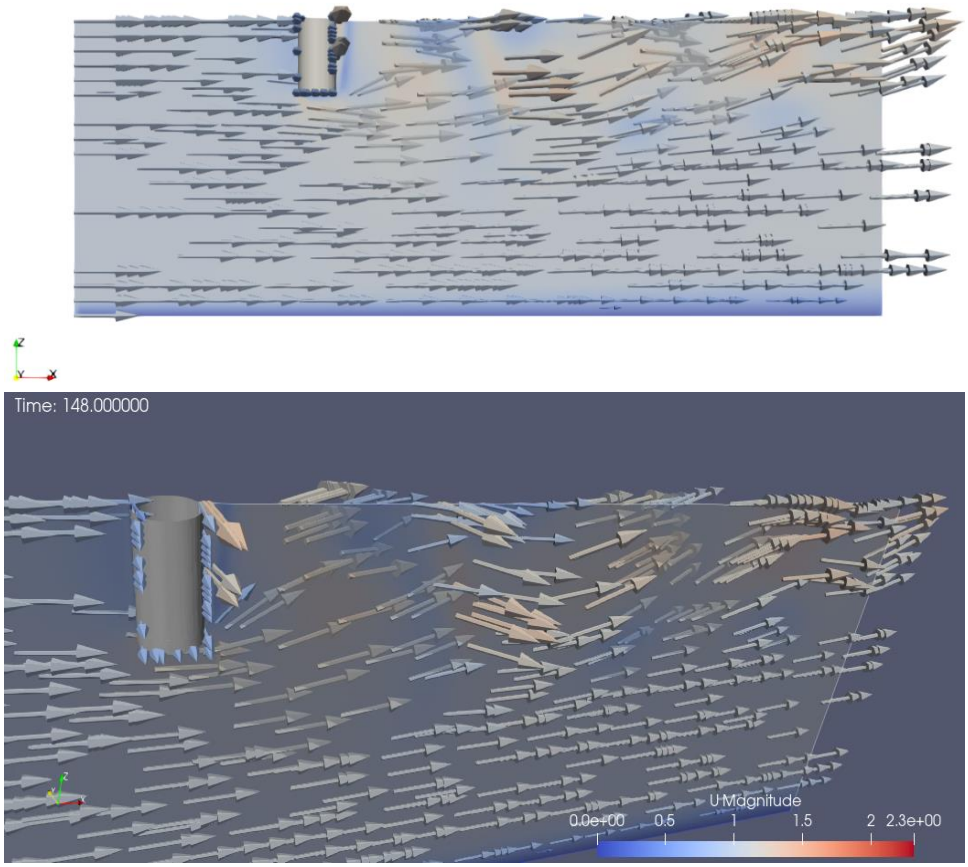
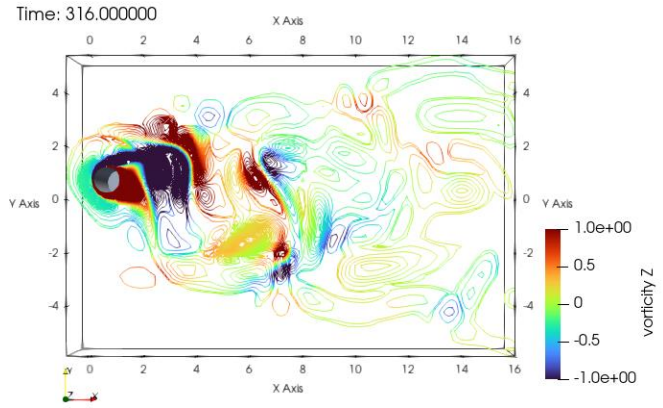
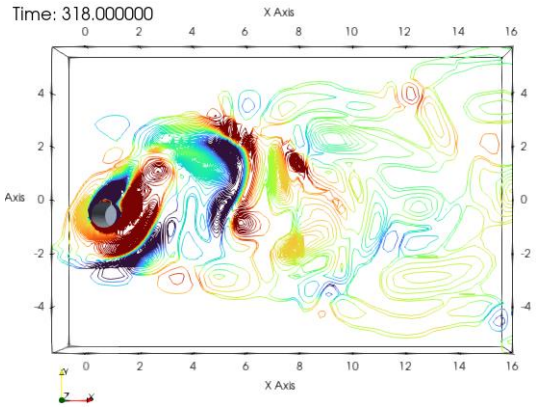


Fig. 5.3.2.34: Vortices in the wake of the cylinder are generated from the free end of the cylinder; 2DOF

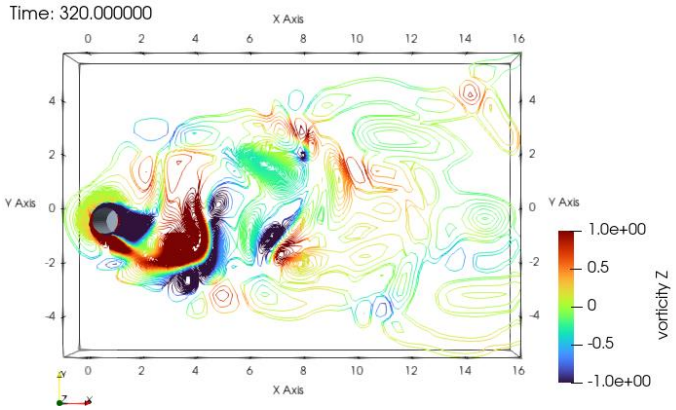
The contours of the non-dimensional vorticity in the axial direction of the cylinder for the 2DOF case on the top surface of the computational domain (z = L) are shown in the Fig. 5.3.2.35 (a-h). Allowing the cylinder to vibrate in the in-line as well as cross-flow direction leads to vortex shedding and vibration at the chosen short cylinder length. The 2D vorticity contours in the axial direction of the cylinder on the top surface of the computational domain is shown in Figs. 5.3.2.35 (a) to (h). The oscillation of the cylinder in both the directions are visible through the axes grids shown.


Time: 316.000000
X Axis
Y Axis
vorticity Z
1.0e+00
0.5
0
-0.5
-1.0e+00

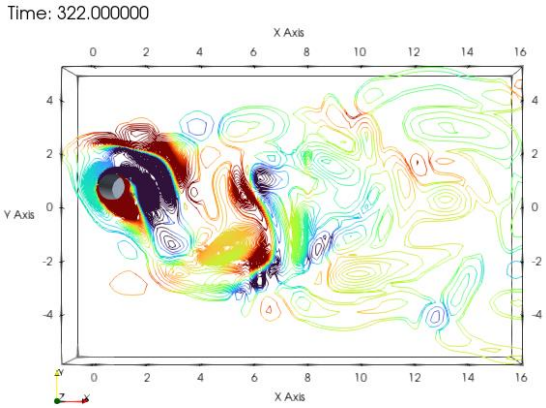
(a)


Time: 318.000000
X Axis

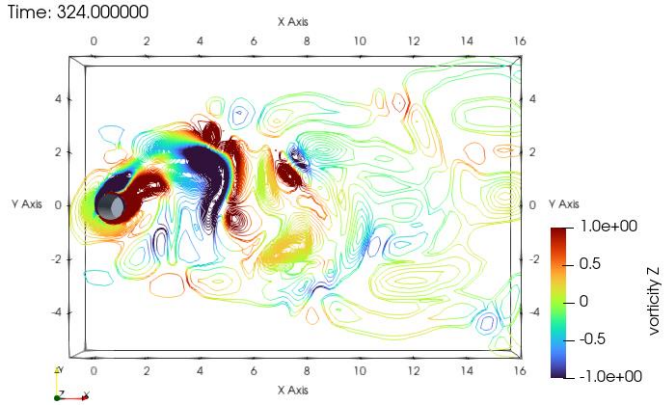
(b)


Time: 320.000000
X Axis
Y Axis
vorticity Z
1.0e+00
0.5
0
-0.5
-1.0e+00

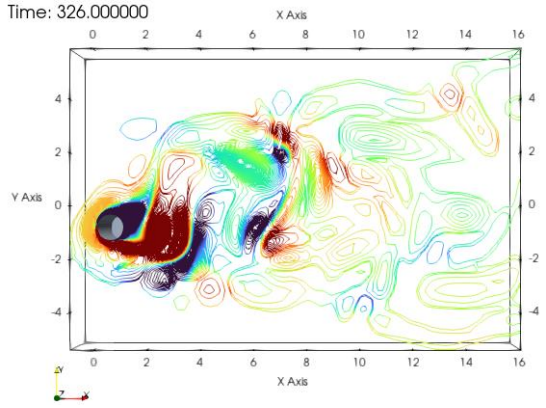
(c)


Time: 322.000000
X Axis
Y Axis

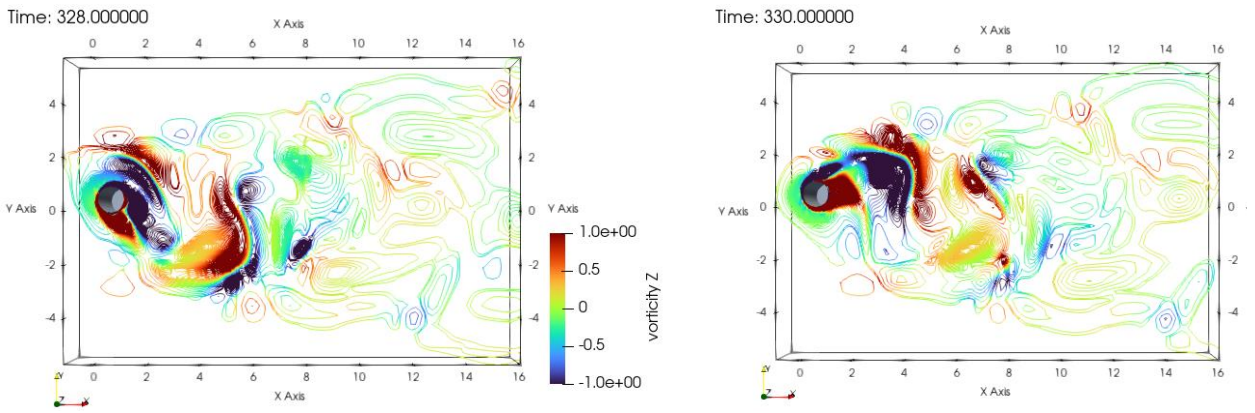
(d)


Time: 324.000000
X Axis
Y Axis
vorticity Z
1.0e+00
0.5
0
-0.5
-1.0e+00

(e)


Time: 326.000000
X Axis
Y Axis

(f)



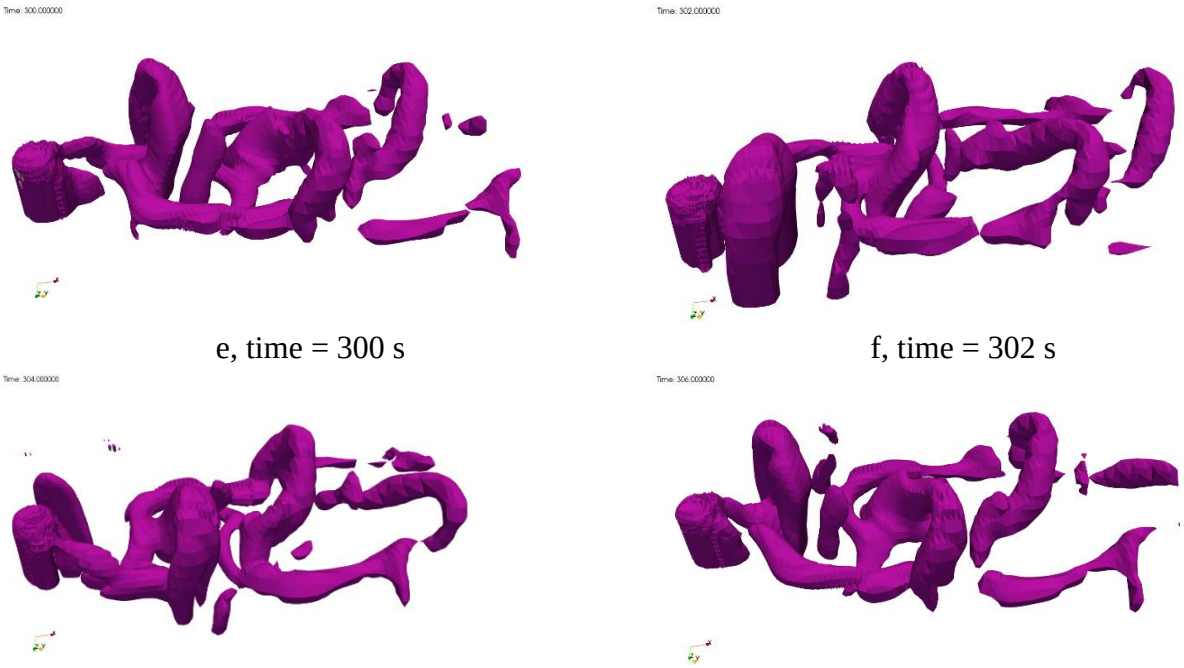
(g) (h)

Fig. 5.3.2.35: Contours of non-dimensional axial vorticity around the cylinder at the top surface of the computation domain for L/D=2 and $V_r = 6$, for the time steps shown; 2DOF

The three-dimensional flow structure in Fig. 5.3.2.36 reveals periodic vortex shedding, that is, the iso-surfaces of the non-dimensional eigenvalue of $e_2 = 0.1$ at 8 instants are visualized from the negative z-direction so as to observe the formation of the vortices near the free end of the cylinder. To demonstate the upwash velocity and that for low aspect ratio the vortices generate more from the ends, the vortex shedding is shown for the inverted cylinder, with the negative z-axis pointing upwards.

a, time = 292 s b, time = 294 s

c, time = 296 s d, time = 298 s



e, time = 300 s f, time = 302 s

g, time = 304 s h, time = 306 s

Fig. 5.3.2.36: Iso-surfaces of the non-dimensional eigenvalue of $e_2 = 0.1$ within half period of vibration for $V_r = 6$visualized from the negative z-direction; 2DOF

A pair of separated shear layer are formed from the two sides of the cylinder, visible in Fig. 5.3.2.36 (g). A pair of vortex tubes (termed "Upper edge vortex" by [52]) originate from the two sides of the upper edge of the cylinder and merge with the shear layers in the downstream, parallel to the incident flow are visible in Fig. 5.3.2.36 (d) and (f). The vortex tube (termed "Lower edge vortex" by [52]) generated from the lower edge of the base column that is moving upwards parallel to the cylinder axis can be easily observed in Fig. 5.3.2.36 (a, d e and f). The base of the cylinder is not visible at any of the cases, as it is covered by a large vortex (termed "Bottom vortex" by [52]). The vortex shedding is visible in Fig. 5.3.2.36 (f to g). Asymmetric and dynamic vortex structure is seen in the wake of the vibrating cylinder. The lift force oscillates as a result of the dynamic vortex flow.

5.3.3 Conclusions

These patterns of vortex generation and vortex shedding gives an idea how the low aspect ratio of the cylinder can contribute to the vortex-induced motion of FOWTs. The single DOF and multi DOF systems do hold a good amount of difference in the vortex forming and shedding patterns. The complete model for any immersed rigid structure exposed to any kind of fluidic force has been developed in this section using OpenFOAM. This model can next be used for any structure, with at least two to all degrees-of-freedom released. The methods to release the available DOFs of a system, in OpenFOAM, is detailed in Annexure-II: Section-4.3.2.

5.4 LARGE EDDY SIMULATIONS (LES) ON FLOW PAST STATIC SQUARE CYLINDER AT REYNOLDS NUMBER 10^5

5.4.1 Problem statement

This part of work is done to get an idea of the response of a square cylinder subjected to turbulent flow. A square cylinder of dimension D, is placed in a fluid domain, with a free stream velocity of 'U0' at zero angle of attack. An inlet turbulence of intensity 0% is set at first, which is the laminar approaching flow case [83]. From this case, the mean and root mean square values of coefficient of drag (CD), have been evaluated and compared with the experimental results [80-82], and presented in Table 1. The schematic of the general problem is shown in Fig. 5.4.1a, and meshing in Fig. 5.4.1b. For the present validation purpose, TLD is not considered.

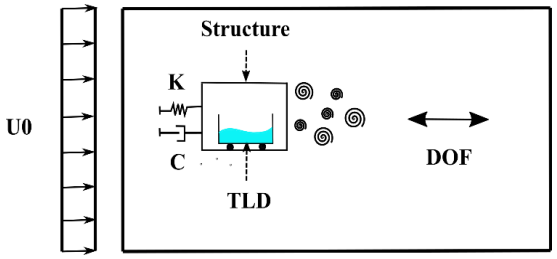


Fig. 5.4.1a. Schematic of the turbulence-structure-TLD system

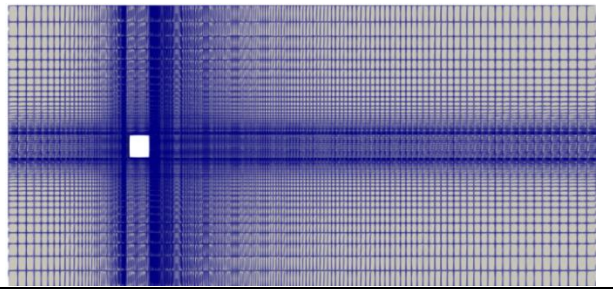


Fig. 5.4.1b. Meshing in OpenFOAM

The entire work is subdivided into two subsections, each consisting of modeling a part of the full solution technique.

a) Modeling the turbulent flow field past a rigid obstacle and estimation of the drag force applied to it.
b) Using the time-varying turbulent forcing to estimate the structural response at each time step with and without attached TLD.

In this chapter, the results of the (a) are shown only, which is an OpenFOAM simulation of flow past a rigid square cylinder subjected to a flow of Reynolds number 10^5. The result and discussion of (b) can be found in Annexure-III: Section-5.4. The fluid considered here is water.

5.4.2 Results and Discussions

Large eddy simulations on static square cylinder are studied at high Reynolds number in order to obtain the drag force on the cylinder face. The Strouhal number, mean drag coefficient and RMS drag coefficient are validated with previous experimental and numerical studies. This is shown in Table: 5.4.1.

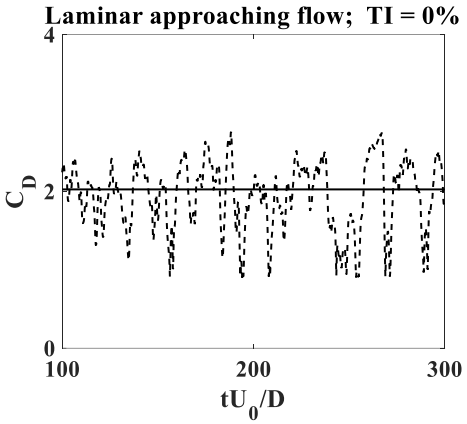


Fig. 5.4.2. Coefficient of drag over time

Table 5.4.1: Validation for flow past square cylinder

	Re	St	$\overline{C_D}$	$\acute{C_D}$
Experimental (Lee 1975) [80]	1.76 x 10^5	0.122	2.07	
Experimental (Pocha 1971) [81]	9.1 x 10^4	0.12	2.06	0.19
Experimental (Noda and Nakayama 2003) [82]	6.89 x 10^4	0.131	2.16	0.207
Numerical (Li et. al 2018) [83]	10^5	0.129	2.085	0.218
Present study	10^5	0.101	2.025	0.206

5.4.3 Conclusions

As seen in Fig. 5.4.2 the turbulent flow is simulated for an initial smaller time duration, which possibly resulted in a lower C_D value. A better prediction is expected for a longer simulation.

The results and discussions of the second part of this work, can be found in Annexure-III: Section-5.4.

CHAPTER 6: CONCLUSIONS

6.1 CONCLUSIONS

In this dissertation vortex-induced vibration (VIV) phenomena are studied. Initially, a few canonical flow problems are solved using in-house MATLAB codes, including a lid-driven cavity and a flow-past static cylinder problems which is extended to a VIV model by attaching a spring to the cylinder. It is then attempted to solve the problem using finite difference approach and estimate the flow-induced vibration behavior of the cylinder. Newmark-β method is employed to solve the non-linear structural motion. The mass and momentum conservation equations of the Navier-Stokes equation, without any explicit pressure equation, is the major limitation. To obtain the values of pressure and velocity at every cell of the flow domain, various semi-implicit, implicit methods have been explored and utilized to solve these equations. The in-house MATLAB codes to solve Navier-Stokes equations have been done using finite difference method and staggered grid configuration. The lid-driven cavity and flow past cylinder cases compared very well with the available results. However, due to high numerical diffusion the VIV model is not carried forward in MATLAB environment. The C++ based open-source flow solver OpenFOAM is opted to solve the problem.

The finite volume based OpenFOAM solver provides a more efficient and robust solution and facilitates to solve for high Reynolds number flow with a very fine computational grid. First, the flow domain around the cylinder is solved. In order to estimate the turbulence Reynolds-averaged Navier Stokes (RANS) equations are solved with $k-\epsilon$ turbulence model. Next, the drag and lift forces are calculated at each time step and used as external force for the cylinder, which is attached to spring system. The Newmark-ß method is used to solve the rigid body motion of the cylinder. The displaced position of the cylinder defines the boundary (no slip wall) at each time step, and the fluid flow changes accordingly. The entire VIV model is then extended to solve the motions of a 3-dimensional offshore spar platform perturbing due to the interacting ocean current. The single degree of freedom is first considered for the validation purpose. Initially the study explored the effect of the number and stiffness of the spring attached to the cylinder. It is observed that at a reduced velocity of 6 ($V_r = 6$) the magnitude of the vortex-induced displacement is the highest. After this, the magnitude decreases. When $Vr = 6$, that is, $Vr < 11$ the vortex shedding frequency and the vibration frequency are same. The lift coefficient and the cylinder displacement are in phase. This is why the vortex shedding in different instances sustains. But, for $Vr > 11$, the vortex shedding frequency and the vibration frequency are the same initially, and the lift coefficient is anti-phase with the cylinder displacement. The anti-phase between the lift coefficient and the vibration frequency does not last. Finally, the 3D spar platform is allowed to move in the two orthogonal directions, i.e., two degrees of freedom system is studied. The time-history of amplitude of the cylinder in the inline and cross-flow direction satisfies the flow physics. In the

1DOF and 2DOF studies, the vortex generation and shedding are observed, which gave a clear idea on the fact that for cylinders with low aspect ratio, the effects of free-end lead to increased amplitude of the cylinder. However, releasing more restraints can give even a better idea. Each case costs 1 central processing unit (CPUs) of 32GB RAM, Intel(R) Xeon(R) W-2133 CPU @ 3.60GHz 3.60 GHz about 108 hours for L/D = 1 and about 168 hours for 2DOF case for the simulated reduced velocity of 6. Thus, a complete 3240 hours (135 days) were required for 1DOF case. Linux as well as Windows Subsystem for Linux (OS Windows 10) have been used.

Finally, the OpenFOAM utility is used to solve a problem of turbulence-induced vibration of a cylinder. Large Eddy Simulation (LES) is used to accurately solve the turbulent flow over a static cylinder with high Reynolds number (10^5). At each time step the drag force around the cylinder is calculated, and due to the force, the cylinder motion is solved.

Thus, in the present work, computational fluid dynamics (CFD) is extensively explored to solve a wide spectrum of problems involving vortex-induced vibration (VIV), using in-house MATLAB codes, and C++ based open-source flow solver, OpenFOAM. The work potentially addressed fully coupled 3D VIV model of offshore spar platform, and further development works can be performed based on the present model.

6.2 SCOPE FOR FUTURE RESEARCH

The present work can serve a good extent to the design of floating offshore wind turbine (FOWT) in terms of displacement of the cylinder at various stiffness of the system. The linear degrees-of-freedom in the inline and cross-flow directions have been kept unrestrained with the help of linear springs. The main setup to simulate vortex-induced vibrations of any structure has been completed in this work, which can contribute to the following in future:

I. The amplitude ratio, rms lift coefficient can be found out for 2DOF VIV case for the range of reduced velocity considered for the 1DOF case.
II. The remaining degrees-of-freedom can be released by using an angular spring, in order to validate with the real scenario of a FOWT.
III. The FOWT can be considered semi-submersible and simulated for multiphase conditions.
IV. The different types of FOWT mentioned in Fig. 1 of Chapter-1 can be simulated.
V. This work can be extended for any civil engineering structure exposed to single-phase fluidic force.

BIBLIOGRAPHY

[1] J. K. , D. K. Jon G., "Structural Analysis," *Structure Magazine, 34: 10-13,* 2016.

[2] R. Faria, S. Oliveira, and A. L. Silvestre, "A Fluid-Structure Interaction Model for Dam-Water Systems: Analytical Study and Application to Seismic Behavior," *Advances in Mathematical Physics,* vol. 2019, pp. 1–14, Jan. 2019, doi: 10.1155/2019/8083906.

[3] F. 2021 IEA: Paris, "IEA, Net Zero by 2050, Technical Report," 2021.

[4] N. DNV GL: Baerum, "DNV GL. Floating Wind: The Power to Commercialize Report;," 2020.

[5] N. DNV: Baerum, "DNV. Floating Offshore Wind: The Next Five Years, Technical Report," 2021.

[6] X. Mei and M. Xiong, "Effects of Second-Order Hydrodynamics on the Dynamic Responses and Fatigue Damage of a 15 MW Floating Offshore Wind Turbine," *J Mar Sci Eng*, vol. 9, no. 11, p. 1232, Nov. 2021, doi: 10.3390/jmse9111232.

[7] M. Lin and Y. Miao, "Importance and Applications of Fluid Dynamics in Civil Engineering and Mechanical Engineering," *Highlights in Science, Engineering and Technology*, vol. 18, pp. 247–252, Nov. 2022, doi: 10.54097/hset.v18i.2681.

[8] A. , & S. Q. Sultana, "Application of Computational Fluid Dynamics in Civil Engineering-A Review," in *National Conference on Recent Advances in Civil Engineering Infrastructure(rACEi-2021)*, 2021.

[9] P. W. Bearman, "Vortex Shedding from Oscillating Bluff Bodies," *Annu Rev Fluid Mech*, vol. 16, no. 1, pp. 195–222, Jan. 1984, doi: 10.1146/annurev.fl.16.010184.001211.

[10] T. Sarpkaya, "A critical review of the intrinsic nature of vortex-induced vibrations," *J Fluids Struct*, vol. 19, no. 4, pp. 389–447, May 2004, doi: 10.1016/j.jfluidstructs.2004.02.005.

[11] C. H. K. Williamson and R. Govardhan, "Vortex-induced vibrations," *Annu Rev Fluid Mech*, vol. 36, no. 1, pp. 413–455, Jan. 2004, doi: 10.1146/annurev.fluid.36.050802.122128.

[12] R. D. Gabbai and H. Benaroya, "An overview of modeling and experiments of vortex-induced vibration of circular cylinders," *J Sound Vib*, vol. 282, no. 3–5, pp. 575–616, Apr. 2005, doi: 10.1016/j.jsv.2004.04.017.

[13] J. Fredsoe and B. M. Sumer, *Hydrodynamics around cylindrical structures (revised edition)*, vol. 26. World Scientific, 2006.

[14] C. H. K. Williamson and R. Govardhan, "A brief review of recent results in vortex-induced vibrations," *Journal of Wind Engineering and Industrial Aerodynamics*, vol. 96, no. 6–7, pp. 713–735, Jun. 2008, doi: 10.1016/j.jweia.2007.06.019.

[15] X. Wu, F. Ge, and Y. Hong, "A review of recent studies on vortex-induced vibrations of long slender cylinders," *J Fluids Struct*, vol. 28, pp. 292–308, Jan. 2012, doi: 10.1016/j.jfluidstructs.2011.11.010.

[16] K.-S. Hong and U. H. Shah, "Vortex-induced vibrations and control of marine risers: A review," *Ocean Engineering*, vol. 152, pp. 300–315, Mar. 2018, doi: 10.1016/j.oceaneng.2018.01.086.

[17] G. Liu, H. Li, Z. Qiu, D. Leng, Z. Li, and W. Li, "A mini review of recent progress on vortex-induced vibrations of marine risers," *Ocean Engineering*, vol. 195, p. 106704, Jan. 2020, doi: 10.1016/j.oceaneng.2019.106704.

[18] Y. Tamura, "Mathematical models for understanding phenomena: Vortex-induced vibrations," *Japan Architectural Review*, vol. 3, no. 4, pp. 398–422, 2020.

[19] C. C. Feng, "The measurement of vortex induced effects in flow past stationary and oscillating circular and D-section cylinders," University of British Columbia, 1968.

[20] M. Zhao and L. Cheng, "Vortex-induced vibration of a circular cylinder of finite length," *Physics of Fluids*, vol. 26, no. 1, p. 015111, Jan. 2014, doi: 10.1063/1.4862548.

[21] C. H. K. Williamson and A. Roshko, "Vortex formation in the wake of an oscillating cylinder," *J Fluids Struct*, vol. 2, no. 4, pp. 355–381, 1988.

[22] R. Govardhan and C. Williamson, "Modes of vortex formation and frequency response of a freely vibrating cylinder," *J Fluid Mech*, vol. 420, pp. 85–130, 2000.

[23] D. Brika and A. Laneville, "Vortex-induced vibrations of a long flexible circular cylinder," *J Fluid Mech*, vol. 250, pp. 481–508, 1993.

[24] A. Khalak and C. H. K. Williamson, "Dynamics of a hydroelastic cylinder with very low mass and damping," *J Fluids Struct*, vol. 10, no. 5, pp. 455–472, 1996.

[25] A. Khalak and C. H. K. Williamson, "Motions, forces and mode transitions in vortex-induced vibrations at low mass-damping," *J Fluids Struct*, vol. 13, no. 7–8, pp. 813–851, 1999.

[26] N. a Jauvtis and C. H. K. Williamson, “The effect of two degrees of freedom on vortex-induced vibration at low mass and damping,” *J Fluid Mech*, vol. 509, pp. 23–62, 2004.

[27] E. Guilmineau and P. Queutey, “Numerical simulation of vortex-induced vibration of a circular cylinder with low mass-damping in a turbulent flow,” *J Fluids Struct*, vol. 19, no. 4, pp. 449–466, 2004.

[28] Z. Y. Pan, W. C. Cui, and Q. M. Miao, “Numerical simulation of vortex-induced vibration of a circular cylinder at low mass-damping using RANS code,” *J Fluids Struct*, vol. 23, no. 1, pp. 23–37, 2007.

[29] M. Zhao and L. Cheng, “Numerical simulation of two-degree-of-freedom vortex-induced vibration of a circular cylinder close to a plane boundary,” *J Fluids Struct*, vol. 27, no. 7, pp. 1097–1110, 2011, doi: https://doi.org/10.1016/j.jfluidstructs.2011.07.001.

[30] P. Anagnostopoulos and G. Iliadis, “Numerical study of the flow pattern and the in-line response of a flexible cylinder in an oscillating stream,” *J Fluids Struct*, vol. 12, no. 3, pp. 225–258, 1998.

[31] S. Mittal and V. Kumar, “Finite element study of vortex-induced cross-flow and in-line oscillations of a circular cylinder at low Reynolds numbers,” *Int J Numer Methods Fluids*, vol. 31, no. 7, pp. 1087–1120, 1999.

[32] S. P. Singh and S. Mittal, “Vortex-induced oscillations at low Reynolds numbers: hysteresis and vortex-shedding modes,” *J Fluids Struct*, vol. 20, no. 8, pp. 1085–1104, 2005.

[33] J. S. Leontini, M. C. Thompson, and K. Hourigan, “The beginning of branching behaviour of vortex-induced vibration during two-dimensional flow,” *J Fluids Struct*, vol. 22, no. 6–7, pp. 857–864, 2006.

[34] Y. Bao, D. Zhou, and J. Tu, “Flow interference between a stationary cylinder and an elastically mounted cylinder arranged in proximity,” *J Fluids Struct*, vol. 27, no. 8, pp. 1425–1446, 2011, doi: https://doi.org/10.1016/j.jfluidstructs.2011.08.008.

[35] M. Zhao, “Flow induced vibration of two rigidly coupled circular cylinders in tandem and side-by-side arrangements at a low Reynolds number of 150,” *Physics of Fluids*, vol. 25, no. 12, p. 123601, Dec. 2013, doi: 10.1063/1.4832956.

[36] E. Wang, W. Xu, X. Gao, L. Liu, Q. Xiao, and K. Ramesh, “The effect of cubic stiffness nonlinearity on the vortex-induced vibration of a circular cylinder at low Reynolds numbers,” *Ocean Engineering*, vol. 173, pp. 12–27, Feb. 2019, doi: 10.1016/j.oceaneng.2018.12.039.

[37] N. Kondo, “Three-dimensional computation for flow-induced vibrations in in-line and cross-flow directions of a circular cylinder,” *Int J Numer Methods Fluids*, vol. 70, no. 2, pp. 158–185, 2012.

[38] D. Lucor, J. Foo, and G. E. Karniadakis, “Vortex mode selection of a rigid cylinder subject to VIV at low mass-damping,” *J Fluids Struct*, vol. 20, no. 4, pp. 483–503, 2005.

[39] S. Mittal and others, “Free vibrations of a cylinder: 3-D computations at Re= 1000,” *J Fluids Struct*, vol. 41, pp. 109–118, 2013.

[40] Navrose and S. Mittal, “Intermittency in free vibration of a cylinder beyond the laminar regime,” *J Fluid Mech*, vol. 870, p. R2, Jul. 2019, doi: 10.1017/jfm.2019.310.

[41] E. Wang, Q. Xiao, and A. Incecik, “Three-dimensional numerical simulation of two-degree-of-freedom VIV of a circular cylinder with varying natural frequency ratios at <math altimg="si97.gif" display="inline" id="mml97" overflow="scroll"> <mi mathvariant="normal">Re</mi> <mo>=</mo> <mn>500</mn> </math>,” *J Fluids Struct*, vol. 73, pp. 162–182, Aug. 2017, doi: 10.1016/j.jfluidstructs.2017.06.001.

[42] D. J. Farivar, “Turbulent uniform flow around cylinders of finite length,” *AIAA journal*, vol. 19, no. 3, pp. 275–281, 1981.

[43] T. Kawamura, M. Hiwada, T. Hibino, I. Mabuchi, and M. Kumada, “Flow around a finite circular cylinder on a flat plate: Cylinder height greater than turbulent boundary layer thickness,” *Bulletin of JSME*, vol. 27, no. 232, pp. 2142–2151, 1984.

[44] S. C. Luo, T. L. Gan, and Y. T. Chew, “Uniform flow past one (or two in tandem) finite length circular cylinder (s),” *Journal of wind engineering and industrial aerodynamics*, vol. 59, no. 1, pp. 69–93, 1996.

[45] C.-W. Park and S.-J. Lee, “Free end effects on the near wake flow structure behind a finite circular cylinder,” *Journal of Wind Engineering and Industrial Aerodynamics*, vol. 88, no. 2–3, pp. 231–246, 2000.

[46] S. C. Roh and S. Park, “Vortical flow over the free end surface of a finite circular cylinder mounted on a flat plate,” *Exp Fluids*, vol. 34, no. 1, pp. 63–67, 2003.

[47] D. Sumner and J. L. Heseltine, “Tip vortex structure for a circular cylinder with a free end,” *Journal of Wind Engineering and Industrial Aerodynamics*, vol. 96, no. 6–7, pp. 1185–1196, 2008.

[48] G. V. Iungo and G. Buresti, “Experimental investigation on the aerodynamic loads and wake flow features of low aspect-ratio triangular prisms at different wind directions,” *J Fluids Struct*, vol. 25, no. 7, pp. 1119–1135, 2009.

[49] G. Palau-Salvador, T. Stoesser, J. Fröhlich, M. Kappler, and W. Rodi, “Large eddy simulations and experiments of flow around finite-height cylinders,” *Flow Turbul Combust*, vol. 84, pp. 239–275, 2010.

[50] D. Sumner, “Flow above the free end of a surface-mounted finite-height circular cylinder: A review,” *J Fluids Struct*, vol. 43, pp. 41–63, Nov. 2013, doi: 10.1016/j.jfluidstructs.2013.08.007.

[51] H. Fukuoka, S. Hirabayashi, and H. Suzuki, “The effects of free surface and end cell on flow around a finite circular cylinder with low aspect ratio,” *J Mar Sci Technol*, vol. 21, pp. 145–153, 2016.

[52] Y. Liu, F. Liu, E. Wang, Q. Xiao, and L. Li, “The effect of base column on vortex-induced vibration of a circular cylinder with low aspect ratio,” *Ocean Engineering*, vol. 196, p. 106822, Jan. 2020, doi: 10.1016/j.oceaneng.2019.106822.

[53] A. Robertson *et al.*, “Definition of the Semisubmersible Floating System for Phase II of OC4,” Golden, CO (United States), Sep. 2014. doi: 10.2172/1155123.

[54] A. Kozakiewicz, B. M. Sumer, and J. Fredsøe, “Spanwise correlation on a vibrating cylinder near a wall in oscillatory flows,” *J Fluids Struct*, vol. 6, no. 3, pp. 371–392, 1992.

[55] A. Kozakiewicz, B. M. Sumer, J. Fredsøe, and E. A. Hansen, “Vortex regimes around a freely vibrating cylinder in oscillatory flow,” *International Journal of Offshore and Polar Engineering*, vol. 7, no. 02, 1997.

[56] M. Zhao and L. Cheng, “Numerical simulation of vortex-induced vibration of four circular cylinders in a square configuration,” *J Fluids Struct*, vol. 31, pp. 125–140, 2012.

[57] M. Zhao, “Numerical investigation of two-degree-of-freedom vortex-induced vibration of a circular cylinder in oscillatory flow,” *J Fluids Struct*, vol. 39, pp. 41–59, 2013.

[58] B. M. Sumer and J. Fredsøe, “Transverse vibrations of an elastically mounted cylinder exposed to an oscillating flow,” 1988.

[59] B. M. Sumer, J. Fredøe, and K. Jensen, “A note on spanwise correlation on a freely vibrating cylinder in oscillatory flow,” *J Fluids Struct*, vol. 8, no. 3, pp. 231–238, 1994.

[60] V. Azadeh-Ranjbar, N. Elvin, and Y. Andreopoulos, “Vortex-induced vibration of finite-length circular cylinders with spanwise free-ends: Broadening the lock-in envelope,” *Physics of Fluids*, vol. 30, no. 10, p. 105104, Oct. 2018, doi: 10.1063/1.5042774.

[61] A. N. Kolmogorov, “Equations of turbulent motion of an incompressible fluid, Izv. Acad. Sci., USSR,” *Physics (College Park Md)*, vol. 6, no. 1, p. 2, 1942.

[62] W. P. Jones and B. E. Launder, “The prediction of laminarization with a two-equation model of turbulence,” *Int J Heat Mass Transf*, vol. 15, no. 2, pp. 301–314, Feb. 1972, doi: 10.1016/0017-9310(72)90076-2.

[63] W. P. Jones and B. E. Launder, “The calculation of low-Reynolds-number phenomena with a two-equation model of turbulence,” *Int J Heat Mass Transf*, vol. 16, no. 6, pp. 1119–1130, Jun. 1973, doi: 10.1016/0017-9310(73)90125-7.

[64] K. Hanjalić and B. E. Launder, “A Reynolds stress model of turbulence and its application to thin shear flows,” *J Fluid Mech*, vol. 52, no. 4, pp. 609–638, Apr. 1972, doi: 10.1017/S002211207200268X.

[65] B. E. Launder and D. B. Spalding, “The numerical computation of turbulent flows,” *Comput Methods Appl Mech Eng*, vol. 3, no. 2, pp. 269–289, Mar. 1974, doi: 10.1016/0045-7825(74)90029-2.

[66] B. E. Launder and B. I. Sharma, “Application of the energy-dissipation model of turbulence to the calculation of flow near a spinning disc,” *Letters in Heat and Mass Transfer*, vol. 1, no. 2, pp. 131–137, Nov. 1974, doi: 10.1016/0094-4548(74)90150-7.

[67] D. C. Wilcox, “Reassessment of the scale-determining equation for advanced turbulence models,” *AIAA Journal*, vol. 26, no. 11, pp. 1299–1310, Nov. 1988, doi: 10.2514/3.10041.

[68] J. C. Kok, “Resolving the dependence on free-stream values for the kx turbulence model—NLR-TP-99295,” 1999.

[69] F. R. Menter, “Influence of freestream values on k-omega turbulence model predictions,” *AIAA Journal*, vol. 30, no. 6, pp. 1657–1659, Jun. 1992, doi: 10.2514/3.11115.

[70] F. R. Menter, “Two-equation eddy-viscosity turbulence models for engineering applications,” *AIAA Journal*, vol. 32, no. 8, pp. 1598–1605, Aug. 1994, doi: 10.2514/3.12149.

[71] D. C. Wilcox and others, *Turbulence modeling for CFD*, vol. 2. DCW industries La Canada, CA, 1998.

[72] P. SPALART and S. ALLMARAS, “A one-equation turbulence model for aerodynamic flows,” in *30th Aerospace Sciences Meeting and Exhibit*, Reston, Virigina: American Institute of Aeronautics and Astronautics, Jan. 1992. doi: 10.2514/6.1992-439.

[73] D. C. Wilcox, “Formulation of the k-w Turbulence Model Revisited,” *AIAA Journal*, vol. 46, no. 11, pp. 2823–2838, Nov. 2008, doi: 10.2514/1.36541.

[74] T.-H. Shih, W. W. Liou, A. Shabbir, Z. Yang, and J. Zhu, “A new k-epsilon eddy viscosity model for high Reynolds number turbulent flows: Model development and validation,” 1994.

[75] M. Leschziner, *Statistical Turbulence Modelling for Fluid Dynamics — Demystified*. IMPERIAL COLLEGE PRESS, 2015. doi: 10.1142/p997.

[76] S. B. Pope, *Turbulent Flows*. Cambridge University Press, 2000. doi: 10.1017/CBO9780511840531.

[77] T. J. Craft, B. E. Launder, and K. Suga, “Development and application of a cubic eddy-viscosity model of turbulence,” *Int J Heat Fluid Flow*, vol. 17, no. 2, pp. 108–115, Apr. 1996, doi: 10.1016/0142-727X(95)00079-6.

[78] C. H. Liu and D. Y. C. Leung, “Development of a finite element solution for the unsteady Navier–Stokes equations using projection method and fractional-θ-scheme,” *Comput Methods Appl Mech Eng*, vol. 190, no. 32–33, pp. 4301–4317, May 2001, doi: 10.1016/S0045-7825(00)00320-0.

[79] M. Zhao, F. Tong, and L. Cheng, “Numerical Simulation of Two-Degree-of-Freedom Vortex-Induced Vibration of a Circular Cylinder Between Two Lateral Plane Walls in Steady Currents,” *J Fluids Eng*, vol. 134, no. 10, Oct. 2012, doi: 10.1115/1.4007426.

[80] B. E. Lee, “The effect of turbulence on the surface pressure field of a square prism,” *J Fluid Mech*, vol. 69, no. 2, pp. 263–282, May 1975, doi: 10.1017/S0022112075001437.

[81] J. J. Pocha, “On unsteady flow past cylinders of square cross-section,” *Ph. D. Thesis, Department of Aeronautics, Queen Mary College*, 1971.

[82] H. Noda and A. Nakayama, “Free-stream turbulence effects on the instantaneous pressure and forces on cylinders of rectangular cross section,” *Exp Fluids*, vol. 34, pp. 332–344, 2003.

[83] Y. C. Li, C. Y. Chung, and F. M. Fang, “Effect of Turbulent Uniform Flow past a Two-Dimensional Square Cylinder,” *Journal of Applied Fluid Mechanics*, vol. 11, no. 5, pp. 1185–1192, Sep. 2018, doi: 10.29252/jafm.11.05.28067.

[84] H. Jasak, “Error analysis and estimation for the finite volume method with applications to fluid flows.,” 1996.

[85] D.C. Wilcox, “The Wilcox k-omega Turbulence Model,” *https://turbmodels.larc.nasa.gov/wilcox.html*, Aug. 2013.

[86] F. R. Menter, “Improved two-equation k-omega turbulence models for aerodynamic flows,” 1992.

[87] F.R. Menter, “The Menter Shear Stress Transport Turbulence Model,” *https://turbmodels.larc.nasa.gov/sst.html*, Jul. 2013.

[88] G. Kalitzin, G. Medic, G. Iaccarino, and P. Durbin, “Near-wall behavior of RANS turbulence models and implications for wall functions,” *J Comput Phys*, vol. 204, no. 1, pp. 265–291, Mar. 2005, doi: 10.1016/j.jcp.2004.10.018.

[89] D. K. Lilly, “On the application of eddy viscosity concept in the inertial sub-range of turbulence,” *NCAR manuscript*, vol. 123, 1966.

[90] S. B. , and S. B. Pope. Pope, *Turbulent flows*. Cambridge university press, 2000.

[91] J. Smagorinsky, “General circulation experiments with the primitive equations: I. The basic experiment,” *Mon Weather Rev*, vol. 91, no. 3, pp. 99–164, 1963.

[92] A. Dullweber, B. Leimkuhler, and R. McLachlan, “Symplectic splitting methods for rigid body molecular dynamics,” *J Chem Phys*, vol. 107, no. 15, pp. 5840–5851, Oct. 1997, doi: 10.1063/1.474310.

[93] L. M. Sun, Y. Fujino, B. M. Pacheco, and P. Chaiseri, “Modelling of tuned liquid damper (TLD),” *Journal of Wind Engineering and Industrial Aerodynamics*, vol. 43, no. 1–3, pp. 1883–1894, Jan. 1992, doi: 10.1016/0167-6105(92)90609-E.

ANNEXURE-I: IN-HOUSE CODING AND MATHEMATICAL DESCRIPTIONS

Annexure-I: Section-3.3

[As a continuation of Section-3, the equations are numbered accordingly. The equation number starts from 3.28 for this section] (Image source: www.fluidmechanics101.com/pages/lectures.html)

Annexure-I: Section-3.3.1

GREEN-GAUSS CELL-BASED AND NODE BASED GRADIENT SCHEMES

Cell centre gradient:

Any vector is of 3 components and in cartesian coordinates it can be defined as

$$\nabla T = \left(\frac{\partial T}{\partial x}, \frac{\partial T}{\partial y}, \frac{\partial T}{\partial z}\right)$$

The need of gradient in CFD code is that it is used in the discretization of convection terms, for:

i) Linear Upwind Differencing
ii) Non-orthogonal correctors
iii) Source terms

The main requirement of gradient is that the face values can be extrapolated from the cell centroid values.

```
gradSchemes
{
    default         Gauss linear;
    grad(U)         cellLimited Gauss linear 1;
}
```

Fig. 3.1: This is an example of defining a gradient for velocity in OpenFOAM

Gradient Calculation

In one-dimensional, average values of the two faces gives the cell centroid value and the gradient across one of the faces of a cell aligned with the x-axis can be given as:

$$(\nabla T)_f = \left(\frac{T_N - T_P}{d_{PN}}, 0, 0\right) \quad [3.28]$$

3D Gradient calculation

There are three main methods to compute the gradient at the cell centroid:

i) Green-Gauss cell based (linear)
ii) Green-Gauss node based (pointLinear)
iii) Least Squares (leastSquares)

The least square method does not use the value at face centers unlike the Green-Gauss method. The notations used in OpenFOAM for the interpolation schemes, present in fvSchemes are shown in Fig. 3.2. These interpolation schemes are thus required only when i) or ii) method is being used.

```
interpolationSchemes
{
    default         linear;
}
```

Fig. 3.2: Notations used in OpenFOAM

The Green-Gauss method follows the Gauss divergence theorem, eq 3.20. The LHS of the equation contains $\nabla . B$ (divergence of B) which is different from ∇B (gradient of B). To calculate the divergence of B, B is considered to be a product of a scalar (T, Temperature) and an arbitrary vector field (here, C).

$$B = T.C \quad [3.29]$$

Deriving from the Gauss divergence theorem,

$$\int_V (\nabla . B) dV = \int_S (B . \hat{n}) dS \quad [3.20]$$

Replacing B as a product of T and C:

$$\int_V (\nabla . TC) dV = \int_S (TC . \hat{n}) dS \quad [3.21]$$

Using product rule in Eq 3.21,

$$\int_V [T(\nabla . C)] dV + \int_V [(\nabla . T)C] dV = \int_S [T(C . \hat{n}) dS + \int_S [(T\hat{n}) . C] dS \quad [3.22]$$

C is set to (1,1,1) which leads to Eq 3.23 here.

$$\int_V (\nabla T) dV = \int_S [(T\hat{n}) . C] dS \quad [3.23]$$

Eq 3.23 is a slight manipulation of the divergence theorem as in the LHS the gradient of the parameter is integrated over the volume, unlike the divergence of the parameter being integrated

similarly. As CFD cells have a finite number of faces (N), the surface integral is replaced with a summation of the surface integrals of each face over the cell which will represent the volume integral of the temperature gradient inside the cell.

$$\int_V (\nabla T) dV = \sum_N \int_S [(T\hat{n}).C] dS \quad [3.24]$$

The face integral is then replaced with the value at the face center (T_f)

In 2nd order FVM, the variation of quantities across the faces is linear. [see right picture], the value of the integral across the face is the same as the value at face center multiplied by the area under it. So, the surface integral can be written as the product of the temperature, normal vector at the face center and area of the face.

$$\int_V (\nabla T) dV = \sum_N [T_f \hat{n}_f A_f] \quad [3.25]$$

Now, to get rid of the integral in LHS, the temperature gradient is taken to be the value at the cell center as the variation across cells is linear.

$$(\nabla T)_P V_P = \sum_N [T_f \hat{n}_f A_f] \quad [3.26]$$

The gradient at the cell centroid of any quantity can be written as:

$$(\nabla T)_P = \frac{1}{V_P} \sum_N [T_f \hat{n}_f A_f];$$
$$(\nabla U)_P = \frac{1}{V_P} \sum_N [U_f \hat{n}_f A_f];$$
$$(\nabla k)_P = \frac{1}{V_P} \sum_N [k_f \hat{n}_f A_f] \text{ and so on} \quad [3.27]$$

Least-squares gradient Scheme

In OpenFOAM, ANSYS Fluent and Star CCM, the flow variables (T, p, U) are calculated and stored in cell centroids. Least square method is one method to calculate the gradient of such flow variables at the cell centroid, besides Gauss Green cell and node-based methods. In OpenFOAM, this can be seen in fvSchemes as shown in Fig. 3.4.

```
gradSchemes
{
    default         none;
    grad(U)         leastSquares;
}
```

Fig. 3.4: gradSchemes in fvSchemes in OpenFOAM

In ANSYS Fluent it can be seen as shown in Fig. 3.5

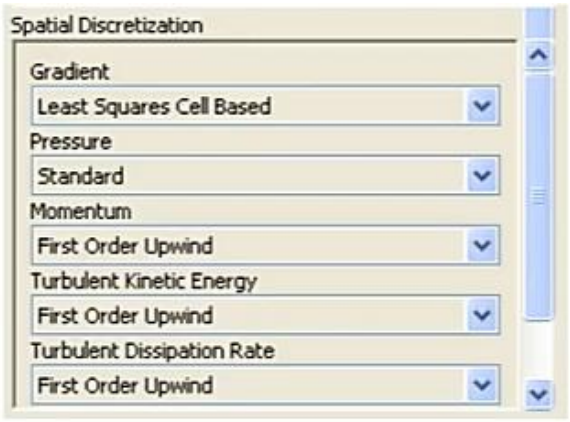


Fig. 3.5: gradient schemes in ANSYS Fluent

As the least squares method does not use any face center values to calculate the gradient of flow variables at the cell centers, so it does not require any interpolation schemes.

Least Squares Derivation

The equation for the neighbor cell across a single face is,

$$T_N = T_p + d_{PN}.(\nabla T)_P \quad [3.30]$$

$(\nabla T)_P$ is the unknown here. An additional equation for every cell face is given by,

$$T_N = T_p + d_{PN}.(\nabla T)_P \quad [3.31]$$

$$T_{N1} = T_p + d_{PN1}.(\nabla T)_P \quad [3.32]$$

$$T_{N2} = T_p + d_{PN2}.(\nabla T)_P \quad [3.33]$$

$$\text{Up to, } T_N = T_p + d_{PN}.(\nabla T)_P \quad [3.34]$$

There is one equation for each cell face. In all of these equations, $(\nabla T)_P$ is the same and is unknown. d_{PN} is the distance between the cell centroid and each of the neighbor cells. Subtracting T_p from both sides of the equations 3.32, 3.33, 3.34,

$$T_{N1} - T_p = d_{PN1}.(\nabla T)_P \quad [3.35]$$

$$T_{N2} - T_p = d_{PN2}.(\nabla T)_P \quad [3.36]$$

$$\text{Up to, } T_N - T_p = d_{PN}.(\nabla T)_P \quad [3.37]$$

From these, a matrix equation can be obtained,

$$[T_N - T_p] = [d_{PN}][(\nabla T)_P] \quad [3.38]$$

$$[d_{PN}][(\nabla T)_P] = [T_N - T_p] \quad [3.39]$$

Eq 3.39 takes the form of $AX = B$

If a quadrilateral cell with N = 4 faces is considered,

$$\begin{bmatrix} d_{PN1,x} & d_{PN1,y} & d_{PN1,z} \\ d_{PN2,x} & d_{PN2,y} & d_{PN2,z} \\ d_{PN2,x} & d_{PN3,y} & d_{PN3,z} \\ d_{PN2,x} & d_{PN4,y} & d_{PN4,z} \end{bmatrix}_{(4\times3)} \begin{bmatrix} (\partial T / \partial x)_P \\ (\partial T / \partial y)_P \\ (\partial T / \partial z)_P \end{bmatrix}_{(3\times1)} = \begin{bmatrix} T_{N1} - T_p \\ T_{N2} - T_p \\ T_{N3} - T_p \\ T_{N4} - T_p \end{bmatrix}_{(4\times1)} \quad [3.40]$$

For N sided cell, $(N \times 3) \times (3 \times 1) = (N \times 1)$

As, the matrices are not square so exact solution is not obtainable. But an approximate solution can be computed for $(\nabla T)_P$ which will lead to an error in each equation.

$$e_1 = T_{N1} - (T_p + (\nabla T)_P.d_{PN1} \quad [3.41]$$

$$e_2 = T_{N2} - (T_p + (\nabla T)_P.d_{PN2} \quad [3.42]$$

The sum of the errors squared can be minimized by the least-squares solution. So, the final equation can be written as,

$$(\nabla T)_P = (d^T d)^{-1} d^T (T_N - T_p) \quad [3.43]$$

If $G = (d^T d)$, then equation 16 can be written as,

$$(\nabla T)_P = G^{-1} d^T (T_N - T_p) \quad [3.44]$$

Now, $G = (d^T d)$ can be written as,

$$G = \begin{bmatrix} d_{PN1,x} & d_{PN2,x} & d_{PN2,x} & d_{PN2,x} \\ d_{PN1,y} & d_{PN2,y} & d_{PN3,y} & d_{PN4,y} \\ d_{PN1,z} & d_{PN2,z} & d_{PN3,z} & d_{PN4,z} \end{bmatrix} \begin{bmatrix} d_{PN1,x} & d_{PN1,y} & d_{PN1,z} \\ d_{PN2,x} & d_{PN2,y} & d_{PN2,z} \\ d_{PN2,x} & d_{PN3,y} & d_{PN3,z} \\ d_{PN2,x} & d_{PN4,y} & d_{PN4,z} \end{bmatrix} \quad [3.45]$$

$$G = \begin{bmatrix} G_{11} & G_{12} & G_{13} \\ G_{21} & G_{22} & G_{23} \\ G_{31} & G_{32} & G_{33} \end{bmatrix} \quad [3.46]$$

Thus, G is always a square matrix and inverse of it is convenient. G^{-1} is required to calculate only once at the instances where the gradient calculation is required for each cell in the mesh, if the mesh is stationary. $(\nabla T)_P$ can then be obtained from a single matrix multiplication. For a static mesh, $G^{-1}\, d^T$ does not change.

But, in the problem chosen, the mesh is moving, where the use of dynamicMeshDict comes into play, the mesh changes at every time instance. To avoid calculation complications, the least square method was not chosen, despite of it being more accurate than the chosen one, here, Gauss-Green cell-based method.

The gradient is re-evaluated if the temperature changes.

$$(\nabla T)_P = \begin{bmatrix} -1/2 & 0 & 1/2 & 0 \\ 0 & -1/2 & 0 & 1/2 \\ 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} T_{N1} - T_p \\ T_{N2} - T_p \\ T_{N3} - T_p \\ T_{N4} - T_p \end{bmatrix} \quad [3.47]$$

For high aspect ratio cells or cells with thin boundary layer, the distance normal to the wall is small. So, the least squares gradient, $(\nabla T)_P$, is dominated by the streamwise gradients, $(T_{Ni} - T_p)$.

Weighted Least-Squares

A weighting function, $[w]$ is introduced in eq 3.39, as

$$[w][d_{PN}][(\nabla T)_P] = [w][T_N - T_p] \quad [3.48]$$

$[w]$ is a diagonal matrix of weights, which can be expressed as, $w_i = 1/|d_i|$

From this, the solution from Eq 3.44 can be written as,

$$(\nabla T)_P = G^{-1}\, d^T W^T W (T_N - T_p) \quad [3.49]$$

$$G = (d^T W^T W d) \quad [3.50]$$

So, for a static mesh Eq 3.50 is evaluated only once.

GREEN-GAUSS NODE-BASED GRADIENT SCHEMES

The gradient selection option in OpenFOAM and ANSYS Fluent is shown below in Fig.3.7.

```
gradSchemes
{
    default         none;
    grad(U)         Gauss pointLinear;
}
```

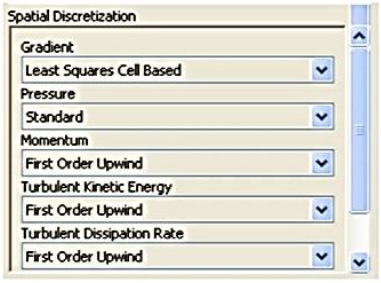


Fig. 3.7: Gradient selection option in OpenFOAM and ANSYS Fluent

This method includes the following steps:

1. The nodal values are calculated from the averaging of the centroid values.	2. The face values are calculated from the nodal values.	3. The gradient is calculated from the face values using Green-Gauss formula
• $T_v = \frac{\sum_{cells} T_P\left(\frac{1}{\lvert d\rvert}\right)}{\sum_{cells}\left(\frac{1}{\lvert d\rvert}\right)}$ [3.51]	• $T_v = \frac{1}{N}\sum_{cells} T_P$ [3.52]	• $(\nabla T)_P = \frac{1}{V_P}\sum_{Faces}[T_f \hat{n}_f A_f]$ [3.53]

The node may not be at the center of the surrounding cells, as shown in Fig. 3.8 and thus requires a weighting function. The weighting function can be defined as, $1/|d|$, where $|d|$ is the distance from the node to the centroid, as written in eq 3.51.

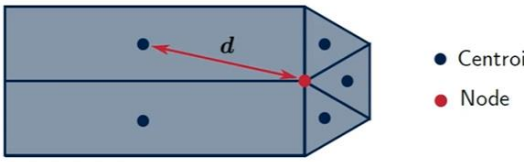


Fig. 3.8: Node not at the center of the surrounding cells

The flow variables are generally constant across the face of the boundary. Hence, the nodal and face center values take the value of the boundary condition.

Mesh Non-Orthogonality

Any two cells of any geometry are to be considered that share a common face. The vector 'd' connects the cell centroid and $\hat{n}$ is the normal vector to the shared face, as shown in Fig. 3.9.

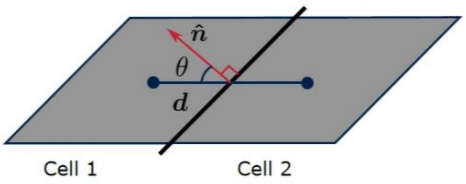


Fig. 3.9: Mesh non-orthogonality for shared face

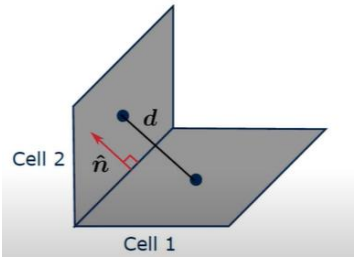


Fig. 3.10.1: Mesh non-orthogonality

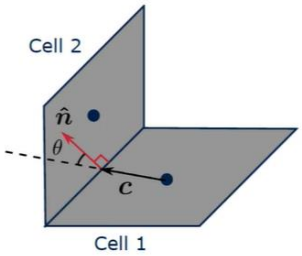


Fig. 3.10.2: Mesh non-orthogonality

If the cells are not aligned, as shown in Fig. 3.10.1, then instead of d, a vector from the cell centroid to face center, 'c', should be used, as shown in Figure right. But the two cells are far from being perfectly orthogonal squares or cubes. θ is calculated in ANSYS Fluent and CFX by using both c and d for all faces in the cell. Then the largest angle is taken as the cell based non-orthogonality (Eq 3.55).

$$\theta_{cell} = \max\left[\cos^{-1}\left(\frac{d.\hat{n}}{|d||\hat{n}|}\right).\cos^{-1}\left(\frac{c.\hat{n}}{|c||\hat{n}|}\right)\right] \qquad [3.55]$$

This is known as orthogonality-quality by ANSYS. θ is normalized to give values between 0 and 1 instead of $0^{\circ}$and 90°. So, the face orthogonality is given more care for computations.

Importance of non-orthogonality in CFD:

Considering the finite-volume discretization of the diffusion term,

$$\frac{\partial U}{\partial t} + \nabla \cdot (UU) = -\frac{1}{\rho}\nabla p + \nabla \cdot (\upsilon \nabla U) + \ g \qquad [3.56]$$

Integrating the diffusion term over the cell volume,

$$\int_V [\nabla . (\nu \nabla U)]\, dV \qquad [3.57]$$

And, using the divergence theorem,

$$\int_V [\nabla . (\nu \nabla U)]\, dV = \int_S [\nu (\nabla U) . \hat{n}]\, dS = \sum_{f=1}^{N} \left[\nu_f (\nabla U)_f \cdot \hat{n}_f\right] S_f \qquad [3.58]$$

In this entire process, the difficulty is in evaluating the dot product of the velocity gradient and the normal vector on the cell face.

$$\nu_f (\nabla U)_f \cdot \hat{n}_f \qquad [3.59]$$

ν_f can be obtained by face interpolation, that is, central differencing. If $\hat{n}$ is parallel to d,

$$(\nabla U)_f \cdot \hat{n}_f = \frac{U_p - U_N}{|d|} |\Delta| \qquad [3.60]$$

According to [84], the unit normal vector should be split into an orthogonal and non-orthogonal component, as shown in Eq 3.61.

$$\hat{n} = \Delta + k \qquad [3.61]$$

Eq 3.61 is then substituted in Eq 3.58,

$$\int_V [\nabla . (\nu \nabla U)]\, dV = \sum_{f=1}^{N} \left[\nu_f (\nabla U)_f \cdot (\Delta + k)\right] S_f \qquad [3.62]$$

$$\int_V [\nabla . (\nu \nabla U)]\, dV = \sum_{f=1}^{N} \left[\nu_f (\nabla U)_f \cdot \Delta_f\right] S_f + \sum_{f=1}^{N} \left[\nu_f (\nabla U)_f \cdot k_f\right] S_f \qquad [3.63]$$

The orthogonal vector, Δ_f can be evaluated implicitly and the non-orthogonal vector, k_f can be evaluated explicitly. Evaluation of the orthogonal term implicitly involves,

$$(\nabla U)_f \cdot \Delta_f = \frac{U_P - U_N}{|d|} \left|\Delta_f\right| \qquad [3.64]$$

This term is calculated implicitly because U_P and U_N are unknowns that get iterated through the momentum and continuity equations.

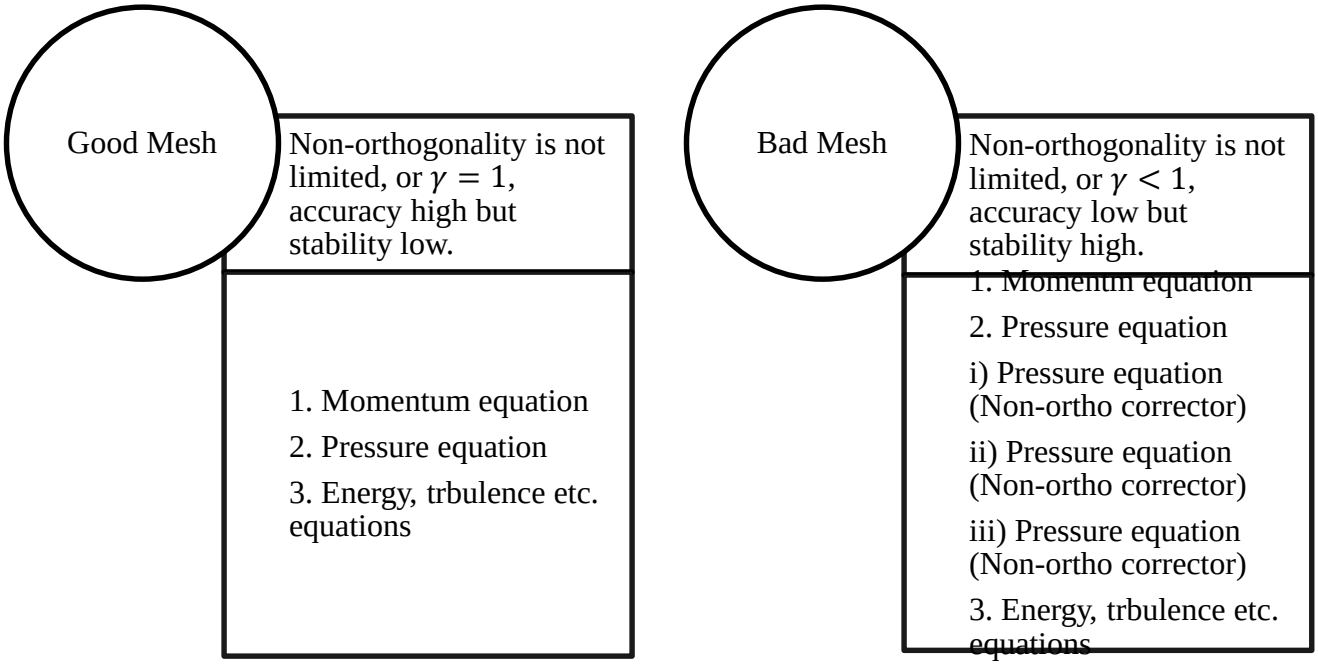
Evaluation of the non-orthogonal term explicitly involves the velocity or parameter that is known, as a source/correction term. Explicit because it is calculated from the known velocity field, U_f, from the previous iteration.

$$(\nabla U)_f \cdot k_f \qquad [3.65]$$

As, this term is not computed implicitly, thus it increases instability. Higher non-orthogonality of mesh means larger explicit term and smaller implicit term. As shown in Figure, $\theta \sim k$, so θ is important and the non-orthogonal correction increases as θ increases. So, a good mesh is required. In OpenFOAM, the non-orthogonal correction is limited as a fraction (γ) of the implicit term,

$$(\nabla U)_f \cdot k_f < \gamma * \left(\frac{U_p - U_N}{|d|} |\Delta_f|\right) \qquad [3.66]$$

The value of γ is generally 1. To improve stability, the accuracy can be compromised and γ can be considered less than 1, but the mesh would not be satisfactory and can be called a bad mesh. Iteration is required because the non-orthogonal corrector uses the existing flow parameter, say, velocity field, U. The outer loops in SIMPLE algorithm are enough for convergence in case of good mesh, that is mesh with non-orthogonality not limited. But in case of bad meshes, additional inner loops of the pressure equation, called non-orthogonal corrector loops are required.



The requirement of explicit non-orthogonal corrector in CFD is to solve the diffusion term. The heat diffusion equation is used to explain this further, where k is the thermal conductivity and T is the temperature.

$$0 = \nabla \cdot (k \nabla T) \qquad [3.67]$$

It is then integrated over cell volume using finite volume method,

$$0 = \int_V \nabla \cdot (k\nabla T) dV \quad [3.68]$$

This is solved using divergence theorem, taking values at face centers.

$$0 = \sum_{Faces} \left[k_f A_f \left(\nabla T_f \cdot \hat{n}_f \right) \right] \quad [3.69]$$

The LHS is zero, since by the law of conservation of energy, the sum of the heat fluxes out of the cell must be zero ($Q = kA\nabla T$). The unknowns here are the temperature at the cell centroids. Thus, ∇T_f should be expressed in terms of cell centroid values, T_P and T_N, in order to calculate them. (Fig. 3.11.1)

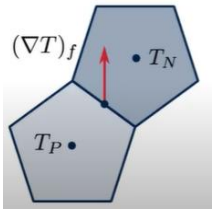


Fig. 3.11.1: Calculation requirement of ∇T_f

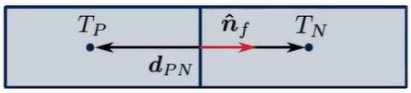


Fig. 3.11.2: Cell centroid values, T_P and T_N,

From Fig. 3.11.2, considering d_{PN} and $\hat{n}_f$ to be parallel,

$$\nabla T_f \cdot \hat{n}_f = \frac{T_N - T_P}{|d_{PN}|} |\hat{n}_f| \quad [3.70]$$

Putting Eq 3.70 in Eq 3.69,

$$\sum_{Faces} \left[k_f A_f \left(\frac{T_N - T_P}{|d_{PN}|} |\hat{n}_f| \right) \right] \quad [3.71]$$

T_P and T_N can be solved by writing one equation for every cell in the mesh and solving the matrices. But for majority of meshes d_{PN} and $\hat{n}_f$ are not parallel, thus the previous method cannot be used. The $\hat{n}_f$ is discretized into $\hat{n}_1$ and $\hat{n}_2$, as

$$\hat{n}_f = \hat{n}_1 + \hat{n}_2$$

$$0 = \sum_{Faces} \left[k_f A_f \left(\nabla T_f \cdot \hat{n}_1 \right) \right] + \sum_{Faces} \left[k_f A_f \left(\nabla T_f \cdot \hat{n}_2 \right) \right] \quad [3.72]$$

$$0 = \sum_{Faces} \left[k_f A_f \left(\frac{T_N - T_P}{|d_{PN}|} |\hat{n}_1| \right) \right] + \sum_{Faces} \left[k_f A_f \left(\nabla T_f \cdot \hat{n}_2 \right) \right] \quad [3.73]$$

The second term in RHS is solved explicitly using the temperature of the previous iteration. This is known as non-orthogonal correction.

UNIT NORMAL VECTOR DECOMPOSITION

Any choice can be made till $\hat{n}_1$ is parallel to the vector joining the cell centroids of the owner and neighboring cells, d_{PN}.

Method-1: Right Angle Triangle (Minimum Correction)

Considering $\hat{n}_2$ perpendicular to $\hat{n}_1$, forming a right-angle triangle, it can be concluded that

$$\hat{n}_1 = \hat{n}_f \cos\theta \qquad [3.74]$$

From the dot product identity, $a.b = |a||b|\cos\theta$, eq 3.74 can be written as,

$$\hat{n}_1 = \hat{n}_f \left(\frac{\hat{n}_f d_{PN}}{|\hat{n}_f||d_{PN}|}\right) \quad \text{and} \quad \hat{n}_2 = \hat{n}_f - \hat{n}_1 \qquad [3.75]$$

θ being the angle between $\hat{n}_1$ and $\hat{n}_f$, it can be seen that as θ increases, $\hat{n}_2$ reduces as much as possible. This limits the size of explicit correction term.

Method-2: Rotation Approach (Orthogonal Correction)

$\hat{n}_f$ is rotated to make it parallel to d_{PN}. The magnitude of $\hat{n}_f$ is taken and multiplied by the direction of d_{PN} (the unit vector in this direction), as

$$\hat{n}_1 = \left|\hat{n}_f\right| \left(\frac{d_{PN}}{|d_{PN}|}\right) \qquad [3.76]$$

Regardless of θ, the magnitude of $\hat{n}_1$ remains the same.

Method-3: Over-Relaxed Approach

$\hat{n}_2$ is assumed to be perpendicular to $\hat{n}_f$.

From Eq 3.75, the magnitude of $\hat{n}_1$ is obtained. As θ increases, the magnitude of $\hat{n}_1$ and $\hat{n}_2$ both increases.

$$\hat{n}_1 = d_{PN} \left(\frac{|\hat{n}_f|^2}{\hat{n}_f d_{PN}}\right) \qquad [3.77]$$

As a comparison between these 3 methods, [84]concluded that all methods give the same solution if sufficient corrector loops are used. The most convenient convergence occurs if over-relaxed method is used. It is the most stable method.

Method-1 and Method-2 diverge when $\theta > 45°$. The over-relaxed approach converges at higher angles, $\theta = 60°$. This is why, the over-relaxed approach is preferred.

For bad meshes, that is, where θ is large, the explicit source term may lead to unboundedness and divergence. So, the explicit source term is limited or completely discarded in case of bad meshes. This leads to local error in bad cells only and thus do not hinder in obtaining a solution. But the error is maximum for Method-3 as $\hat{n}_2$ is the largest.

Annexure-I: Section-3.4

Reynolds Averaged Navier-Stokes Equations

[*The equation number starts from 3.2.9 for this section*]

The RANS equation in tensor notation can be written as:

$$\frac{\partial(\rho U_i)}{\partial t} + \frac{\partial(\rho U_i U_j)}{\partial x_j} = -\frac{\partial P}{\partial x_i} + \frac{\partial}{\partial x_j}\left[\mu\left(\frac{\partial U_i}{\partial x_j} + \frac{\partial U_i}{\partial x_j}\right) - \rho\overline{u_i' u_j'}\right] \quad [3.2.9]$$

Where, 'U' is the mean flow velocity (m/s) in i^{th} or j^{th} directions (2D), 'P' is the pressure, 't' is time (s), ρ is the fluid density (kg/m^3), μ is the absolute viscosity of the fluid (Pa-s). The Reynolds stresses are solved using the Boussinesq constitutive relation [18] or Eddy-viscosity model,

$$-\rho\overline{u'v'} = \mu_t \frac{\partial U}{\partial y} \quad [3.2.10]$$

μ_t is a fictitious quantity for which the modeling is shifted from Reynolds stress tensor to eddy viscosity (μ_t), thus the naming. Further, it controls the strength of diffusion, that is, more the μ_t, more is the transfer of momentum from faster to slower-moving fluid particles.

As the order of multiplication does not matter, so the shear stress components can be written as such:

$$-\overline{\rho u'v'} = -\rho\overline{v'u'} = \mu_t\left(\frac{\partial U}{\partial y} + \frac{\partial V}{\partial x}\right) \quad [3.2.11]$$

This is why the Reynolds stress tensor is symmetric and there are six independent components. Eq. 3.2.10 and Eq. 3.2.11 are the diagonal terms of the Reynolds stress tensor, but for the normal components, Eq. 3.2.11 becomes,

$$-\rho\overline{u'v'} = 2\mu_t \frac{\partial U}{\partial y} \quad [3.2.12]$$

Now, by the definition of turbulent kinetic energy (k), it is the sum of the Reynolds stress terms. But, by adding the Reynolds stress terms, an inconsistent solution of turbulent kinetic energy is obtained, as shown below:

$$-\rho(\overline{u'u'} + \overline{v'v'} + \overline{w'w'}) = 2\mu_t \left(\frac{\partial U}{\partial y} + \frac{\partial V}{\partial x} + \frac{\partial W}{\partial z}\right) \quad [3.2.13]$$

$$\text{As, } k = \frac{1}{2}(\overline{u'u'} + \overline{v'v'} + \overline{w'w'})$$

So, sum of the normal components of Reynolds stress terms should be

$$-\rho(\overline{u'u'} + \overline{v'v'} + \overline{w'w'}) = 2\rho k \quad [3.2.14]$$

which is not equal to Eq. 3.2.13. Again, for an incompressible flow, Eq. 3.2.12 and Eq. 3.2.13 become inconsistent, because

$$\frac{\partial U}{\partial y} + \frac{\partial V}{\partial x} + \frac{\partial W}{\partial z} = 0 \quad [3.2.15]$$

Therefore 1/3rd of the sum of the normal components is subtracted from the over predicting error Reynolds stress term giving

$$-\rho\overline{u'u'} = 2\mu_t \left\{\frac{\partial U}{\partial x} - \frac{1}{3}\left(\frac{\partial U}{\partial y} + \frac{\partial V}{\partial x} + \frac{\partial W}{\partial z}\right)\right\} - \frac{1}{3}(2\rho k) \quad [3.2.16]$$

$$-\rho\overline{v'v'} = 2\mu_t \left\{\frac{\partial V}{\partial x} - \frac{1}{3}\left(\frac{\partial U}{\partial y} + \frac{\partial V}{\partial x} + \frac{\partial W}{\partial z}\right)\right\} - \frac{1}{3}(2\rho k) \quad [3.2.17]$$

$$-\rho\overline{w'w'} = 2\mu_t \left\{\frac{\partial W}{\partial z} - \frac{1}{3}\left(\frac{\partial U}{\partial y} + \frac{\partial V}{\partial x} + \frac{\partial W}{\partial z}\right)\right\} - \frac{1}{3}(2\rho k) \quad [3.2.18]$$

This is done, so that the summation of these equations, that is, the normal components justify the turbulent kinetic energy. Writing Eq. 3.2.11, Eq. 3.2.16, Eq. 3.2.17 and Eq. 3.2.18 combined in tensor notation, in terms of mean rate of strain tensor (S_{ij}) and it's deviatoric part ($S_{ij}{}^*$)

$$-\rho\overline{u_i'u_j'} = 2\mu_t \left(S_{ij} - \frac{1}{3}\frac{\partial U_k}{\partial x_k}\delta_{ij}\right) - \frac{2}{3}\rho k\delta_{ij} \quad [3.2.19]$$

$$\text{and,} \quad -\rho\overline{u_i'u_j'} = 2\mu_t S_{ij}{}^* - \frac{2}{3}\rho k\delta_{ij} \quad [3.2.20]$$

Shear stresses can be written as

$$\text{Mean rate of strain tensor: } S_{ij} = \mu_t \left(\frac{\partial U_i}{\partial x_j} + \frac{\partial U_j}{\partial x_i}\right) \quad [3.2.21]$$

Where, the repeated indices (k, k) represent summation and δ_{ij} is the Kronecker delta,

δ_{ij}= {1 $for\ i = j\ and$ 0 $for\ i \neq j$

In OpenFOAM, Eq. 3.2.20 is depicted as:

$$-\rho\overline{\acute{u}\acute{u}} = \mu_t[\nabla U + (\nabla u)^T] - \frac{1}{3}(\nabla U)I - \frac{2}{3}\rho kI \qquad [3.2.22]$$

Where, δ_{ij}= {1 $for\ \ i = j\ and$ 0 $for\ \ i \neq j$ = $\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$

Before the k - ε model, there were models that used the concept of mixing length, which represents the range of size of the eddies or the turbulence structure that exist in the flow. More energetic the flow is, more the mixing length is and it can be formulated as:

$$\mu_t = \rho k^{1/2} l_m \quad \text{or} \quad \mu_t = \rho l_m^2 \left|\frac{\partial U}{\partial y}\right|$$

This is further specified in Prandtl mixing length hypothesis, which says that the mixing length of the eddies are affected at a distance of 'y' due to the presence of the wall,

$$l_m = \kappa y; \quad \kappa = 0.41$$

But, not only the presence of a wall but the viscosity in the viscous sub-layer also impacts the mixing length by damping the eddies and reducing their size. An account of this damping can be given by the Van Driest model, which says,

$$l_m = \kappa y\left[1 - \exp\left(-\frac{y^+}{A^+}\right)\right]; \qquad A+ = 26.0$$

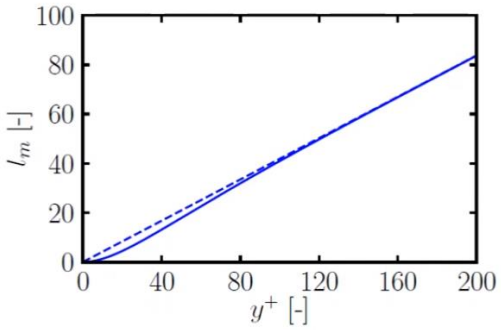


Fig. 3.2.1: Prandtl mixing length vs van Driest model mixing length

The dashed line represents the linear Prandtl model and the solid line represents the Van Driest model which shows that the mixing length reduces as the wall is approached. The mixing length is actually specified algebraically, which indicates that the distance of the wall is specified throughout the entire domain. But, turbulent fluid encounters convection and diffusion so instead of the mixing length, a transport Equation is solved. This is when the transport equation for

turbulent kinetic energy and the turbulence dissipation rate comes into action and thus the k – ε model.

1) $\boldsymbol{k-\varepsilon}$ Model: [65]

$$\mu_t = \rho C_\mu \frac{k^2}{\varepsilon}; \qquad C_\mu = 0.09 \qquad [3.2.23]$$

The Reynolds stress in the RANS equation need to be modelled to close the equations. This can be written in vector notation as:

$$\frac{\partial(\rho U)}{\partial t} + \nabla.(\rho UU) = -\nabla p + \nabla.\mu[(\nabla U + (\nabla U)^T] + \rho g - \nabla\left(\frac{2}{3}\mu(\nabla.U)\right) - \nabla.(\rho\overline{U'U'}) \quad [3.2.24]$$

$$\text{or, } \frac{\partial(\rho U)}{\partial t} + \nabla.(\rho UU) = -\nabla p + \nabla.(\mu + \mu_t)[((\nabla U + (\nabla U)^T)] - \frac{2}{3}\mu(\nabla.U)I] + \rho g \quad [3.2.25]$$

The mixing length can be calculated from the turbulence dissipation rate by:

$$l_m = \frac{C_\mu k^{3/2}}{\epsilon} \qquad [3.2.26]$$

The transport Equation for turbulence kinetic energy is same for standard, RNG and realizable k – ε model,

$$\frac{\partial(\rho k)}{\partial t} + \nabla.(\rho Uk) = \nabla.\left[(\mu + \frac{\mu_t}{\sigma_k})\nabla k\right] + \mathrm{P}_k + \mathrm{P}_b - \rho\epsilon + \mathrm{S}_k \qquad [3.2.27]$$

$\frac{\partial(\rho k)}{\partial t}$: Time

$\nabla.(\rho Uk)$: Convection

$\nabla.\left[(\mu + \frac{\mu_t}{\sigma_k})\nabla k\right]$: Diffusion

$\mathrm{P}_k + \mathrm{P}_b - \rho\epsilon + \mathrm{S}_k$: Sources and sink, where P_k = Production due to mean velocity shear; P_b = Production due to buoyancy; S_k = User-defined source

The transport equation for turbulence dissipation rate is,

$$\frac{\partial(\rho\epsilon)}{\partial t} + \nabla.(\rho U\epsilon) = \nabla.\left[(\mu + \frac{\mu_t}{\sigma_k})\nabla\epsilon\right] + \mathrm{C}_1\frac{\epsilon}{k}(\mathrm{P}_k + \mathrm{C}_3\mathrm{P}_b) - \mathrm{C}_2\rho\frac{\epsilon^2}{k} + S_\epsilon \qquad [3.2.28]$$

$\frac{\partial(\rho\epsilon)}{\partial t}$: Time

$\nabla.(\rho U\epsilon)$: Convection

$\nabla.\left[(\mu + \frac{\mu_t}{\sigma_\epsilon})\nabla\epsilon\right]$: Diffusion

$C_1 \frac{\epsilon}{k}(P_k + C_3 P_b) - C_2\rho\frac{\epsilon^2}{k} + S_\epsilon$: Sources and sink

C_1, C_2 and C_3 depends on the selection of the k – ε model. Comparing the two transport equations, eddy viscosity can be obtained eq 3.2.23.

The model coefficients are present in many journal papers which has evolved consequently and are used in various CFD software.

In mixing length model, the Van Driest approach depicted the damping of the turbulence dissipation rate unlike the k – ε model, where the damping functions (f_1, f_2, f_μ) damps the model coefficients (C_1, C_2, C_μ) to damp or reduce the dissipation rate close to the wall. This means that the Equations of k – ε model can be applied up to the wall, that is, even when the first cell from the wall is present in the viscous sub-layer (y+ < 5). This is known as low-Re formulation.

The damping functions are:

$$f_1 = 1$$

$$f_2 = 1 - 0.3\exp(Re_T^2) \qquad [3.2.29]$$

$f_\mu = exp\left(\frac{-3.4}{(1+(Re_T/50))^2}\right)$ and, Re_T = Turbulent Reynolds number $= \frac{\rho k^2}{\mu\epsilon}$

$$Re_T = \frac{\rho k^{1/2} l_m}{\mu} \sim \left(\frac{\rho U L}{\mu}\right)$$

Replacing the mixing length:

$$Re_T = \frac{\rho k^2}{\mu\epsilon} \qquad [3.2.30]$$

In low Re formulation, the eddy/turbulent viscosity is computed from k and epsilon as,

$$\mu_t = f_\mu C_\mu \frac{\rho k^2}{\epsilon} \qquad [3.2.31]$$

Large Re_T means turbulent forces are dominating and thus it means the point of consideration is far away from the wall. On the other hand, very small Re_T means molecular viscosity is dominating

the turbulent viscosity close to the wall. So, the laminar viscosity will dominate the diffusion term in the momentum equations.

$$\ldots + \nabla.[(\mu + \mu_t)(\nabla U + (\nabla U)^T$$

The damping functions are not point specific values rather applied to every cell in the mesh and it is less than 1 near the wall and increases on moving further away from the wall.

The damping function f_1 was found to be not making any considerable improvement and thus is considered to be 1.

$$\frac{\partial(\rho\epsilon)}{\partial t} + \nabla.(\rho U\epsilon) = \nabla.\left[\left(\mu + \frac{\mu_t}{\sigma_\epsilon}\right)\nabla\epsilon\right] + C_1\frac{\epsilon}{k}(f_1 \mathrm{P}_k + \mathrm{C}_3\mathrm{P}_b) - \mathrm{C}_2\rho\frac{\epsilon^2}{k} + S_\epsilon \qquad [3.2.32]$$

The damping function f_2 however applies more to the dissipation of turbulent dissipation rate:

$$\frac{\partial(\rho\epsilon)}{\partial t} + \nabla.(\rho U\epsilon) = \nabla.\left[\left(\mu + \frac{\mu_t}{\sigma_\epsilon}\right)\nabla\epsilon\right] + C_1\frac{\epsilon}{k}(f_2 \mathrm{P}_k + \mathrm{C}_3\mathrm{P}_b) - \mathrm{C}_2\rho\frac{\epsilon^2}{k} + S_\epsilon \qquad [3.2.33]$$

The k – ε model is useful to resolve the mesh at the viscous sub-layer, that is, near the wall, at low Reynolds number, like that in case of knowing wall shear stress and heat transfer applications, where y^+ values are less than 5 or 1. This is done in external aerodynamic simulation or turbomachinery simulations. But over the years it has been found that the k - ω SST model is better at resolving the mesh at y^+ less than 5, than the k - ε model. So, the k - ε model is used in high Re applications with y^+ > 30, for which the high Re formulations are required to be solved and there are no requirements of damping functions.

Table-3.1: Evolution of the model coefficients of the standard $k - \epsilon$ model

Model	σ_k	σ_ϵ	C_1	C_2	C_3
Jones & Launder [62]	1.0	1.3	1.55	2.0	0.09
Launder & Spalding [65]	1.0	1.3	1.44	1.92	0.09
Launder & Sharma [66]	1.0	1.3	1.44	1.92	0.09

2) k - ω model: [85]

The Eqns. 23 and 25 have been solved by this new closure model. Generic meaning of the terms ε and ω:

ε is the turbulent dissipation rate, the rate at which turbulent kinetic energy is converted into thermal energy by viscosity. The plot of turbulent energy cascade describes the amount of energy contained in different size of the eddies. This plot explains that all of the turbulent kinetic energy is contained when there are large eddies and it is dissipated when there are small eddies.

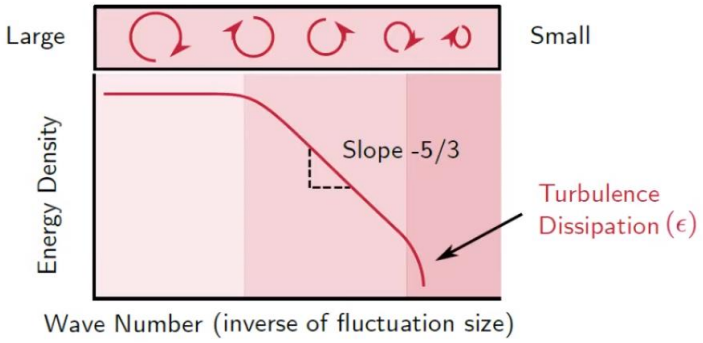


Fig. 3.2.2: Turbulent energy cascade

$$\epsilon = \nu \overline{\frac{\partial U_i'}{\partial x_j} \frac{\partial U_i'}{\partial x_j}} \qquad [3.2.34]$$

The fluctuations $(U_i{}')$ are unknown, so ε cannot be directly found out from Eqn. 34. So, transport Equations, Eqn. 35 is solved to calculate ε.

$$\frac{\partial(\rho k)}{\partial t} + \nabla \cdot (\rho k U) = \nabla \cdot \left(\left(u + \frac{u_t}{\sigma_k} \right) \nabla k \right) + P_\epsilon - \rho\epsilon \qquad [3.2.35]$$

In Eqn. 35, the ε has a negative sign before it indicating that it acts as a sink where the turbulent kinetic energy is dissipated. In practical, there is high dissipation near walls, shear layer and where turbulence is going to be high. ω is another representation of dissipation of turbulence, so often known as a specific turbulence dissipation rate, given by:

$$\omega = \frac{\epsilon}{C_\mu k}; \qquad C_\mu = 0.09 \qquad [3.2.36]$$

This model is detailed in the work by [67]. The transport Equation for calculating ω,

$$\frac{\partial(\rho\omega)}{\partial t} + \nabla.(\rho\omega U) = \nabla.((\mu + \frac{\mu_t}{\sigma_k})\nabla\omega) + \frac{\gamma}{\nu_t} P_k - \beta\rho\omega^2 \qquad [3.2.37]$$

The main difference between the above mentioned two turbulence models is that in k - ω model there are various empirical coefficients depending on the form of model being used. Choice of these coefficients in different solvers may alter the end results. Although both the models are similar, k - ω model is better for aerodynamics and turbomachinery which the k - ε model is not,

as explained earlier. This is because in $k-\varepsilon$ model the damping functions (f_1, f_2, f_μ) used are not that accurate in the presence of adverse pressure gradients. But k - ω model does not need these damping functions when adverse pressure gradient is present. The limitations of this model are explained in Literature review, as mentioned in [68], that this model is dependent on free-stream turbulence conditions.

The remedy for this limitation of both the models was proposed [86]. A blend of both the models is used by using the $k-\varepsilon$ model far away from the wall in the free stream when it is not susceptible to small changes in k and ω; and then near the wall the k - ω model is used. In between these, a blend of both the models is used which forms the basis of the k - ω SST (1992) model.

3) k - ω BST and SST model: [87]

These models are explained in detail in [85]. ε in the transport Equations of $k-\varepsilon$ model, Eqn. 3.2.27 and Eqn. 3.2.28 is substituted with ω from the relation, Eqn. 3.2.38 and Eqn. 3.2.39 is obtained which is same as the transport Equations of k - ω model Eqn. 3.2.37, but with an additional term.

$$\epsilon = \mathrm{C_\mu k\omega} \qquad [3.2.38]$$

$$\frac{\partial(\rho\omega)}{\partial t} + \nabla.(\rho\omega U) = \nabla.\left(\left(\mu + \frac{\mu_t}{\sigma_k}\right)\nabla\omega\right) + \frac{\gamma}{\nu_t} P_k - \beta\rho\omega^2 + 2\frac{\rho\sigma_{\omega^2}}{\omega}\nabla\mathrm{k}:\nabla\omega \qquad [3.2.39]$$

A blending function, $(1-F_1)$ is chosen which builds a relation between the two models,

$$2(1-F_1)\frac{\rho\sigma_{\omega^2}}{\omega}\nabla\mathrm{k}:\nabla\omega \qquad [3.2.40]$$

$$\nabla\mathrm{k}:\nabla\omega = \frac{\partial \mathrm{k}}{\partial \mathrm{x_j}}\frac{\partial\omega}{\partial \mathrm{x_j}} = \frac{\partial \mathrm{k}}{\partial \mathrm{x}}\frac{\partial\omega}{\partial \mathrm{x}} + \frac{\partial \mathrm{k}}{\partial \mathrm{y}}\frac{\partial\omega}{\partial \mathrm{y}} + \frac{\partial \mathrm{k}}{\partial \mathrm{z}}\frac{\partial\omega}{\partial \mathrm{z}} \qquad [3.2.41]$$

The model is $k-\varepsilon$, if $F_1 = 0$ and the model is k - ω, if $F_1 = 1$. Thus, the Eqn. 27 and Eqn. 39 form the transport Equations for this model. This is known as the k - ω Baseline Stress Transport (BST) model. The blending function F_1 is considered one at the cells nearest to the wall and its value decreases to 0 through the farther cells. F_1 is given by:

$$F_1 = \tanh(arg_1^4) \qquad [3.2.42]$$

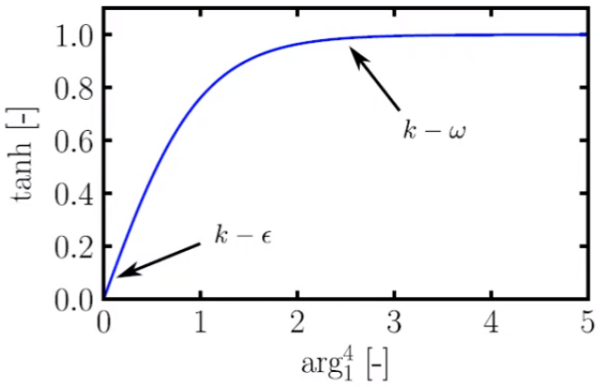


Fig. 3.2.3: Transition between two models

The hyperbolic tan is used so that there is a smooth transition between the two models. The argument arg_1 depends on the distance, 'd' closest to the wall, which is sometimes same as the wall normal distance 'y'.

$$arg_1 = min\left[\max\left(\frac{\sqrt{k}}{\beta^*\omega d}\frac{500\nu}{d^2\omega}\right), \frac{4\rho\sigma_{\omega 2}k}{CD_{k\omega}d^2}\right] \qquad [3.2.43]$$

From the relation, it is clear that when 'd' is small arg_1 will be large and thus F_1 will tend to be 1, that is, near to the wall. [This 'd' is specified at the beginning for a stationary mesh, but for a moving mesh 'd' is to be specified for each computation or change in position.]

F1 not only blends the Equations together but also blends the empirical constants of the two models and that is done by,

$$\phi = F_1\phi_\omega + (1 - F_1)\phi_\epsilon \qquad [3.2.44]$$

ϕ_ω is the model constant (β^*) in k - ω model and ϕ_ϵ (C_μ) in $k-\varepsilon$ model.

4) k - ω Shear Stress Transport (SST) model: [86]

The k - ω BST model was found to be overpredicting the wall shear stress, so the BST model was extended into the SST model with a viscosity limiter [70].

$$\text{Original: } \mu_t = \frac{\rho k}{\omega}; \qquad \text{SST Model: } \mu_t = \frac{a_1\,\rho k}{max\,(a_1\omega, SF_2)} \qquad [3.2.45]$$

The purpose is to limit the viscosity thus reduce the wall shear stress to a more accurate level, closer to the experimental measurements of separated flow. 'S' is the magnitude of shear strain. F_2 is another blending function, which if large then viscosity is reduced.

$$F_2 = tanh\,(arg_2^2) \quad [3.2.46]$$

$$arg_2 = max\,(\frac{2\sqrt{k}}{\beta^*\omega d}, \frac{500\nu}{\omega d^2}) \quad [3.2.47]$$

'd' is same as that for F_1 but can never be the wall normal distance 'y'. The main difference between the original model, BST model and SST model is expressed in Eqn. 3.2.45 and also lies in the formulation of production term (P_ω) of ω in the SST model.

The production term, P_ω: $$P_\omega = \gamma \frac{\omega}{k} \tau_{ij} \cdot \frac{\partial u_i}{\partial x_j} \quad [3.2.48]$$

The dimensionless group, $\nu_t \left(= \frac{\omega}{k}\right)$ is introduced in front of the strain rate tensor which is equal to one in the original and BST models. It had been concluded that k - ω SST model is in better agreement to experiments for mildly separated flows, that is, external aerodynamics or simulations where separation is required. This model has undergone improvements and changes since it's development and it is updated in [87].

5) Spalart Allmaras Model [72]

This model was proposed by Spalart and Allmaras [72] as an improvement over the k - ε model, to build a model that can simulate boundary layer for adverse pressure gradients. It is stated in [88] that, the profile of ν_t near the wall of a flat plate, in the log-law region ($y^+ > 30$) is linear and in the viscous sub-layer ($y^+ < 5$), the profile is quartic, i.e., varies with $(y^+)^4$.

To resolve the quartic variation, the mesh needs to be quite fine close to the wall, as flow quantities vary linearly through cells in CFD. So, to solve this incongruity, instead of ν_t, a similar variable, the Spalart-Allmaras variable called $\tilde{\nu}$ is solved in order to make the solution more stable and easier with a smaller number of cells closer to the wall. $\tilde{\nu}$ varies linearly between ν_t and y+. This linear profile is most suitable for flow over flat plates. For boundary layer over a flat plate with zero pressure gradient,

$$\tilde{\nu} = \kappa y^+ \quad [3.2.49]$$

Other than the case of a flat plate, the likely-to-be-linear profile would not be sufficient for any solution by the CFD code. This is when the transport equation for $\tilde{\nu}$ is solved and from that the ν_t is obtained (Eqn. 50) and fetched into the momentum equations.

$$\nu_t = \tilde{\nu} f_{v1} \qquad \text{where, } f_{v1} = \frac{\chi^3}{\chi^3 + c_{v1}^3} \quad \text{and } \chi = \frac{\tilde{\nu}}{\nu} \quad [3.2.50]$$

The linear profile near to a wall for finite Re is explained in [72]. The transport Equation for the Spalart-Allmaras variable, $\tilde{\nu}$, is given by,

$$\frac{\partial \tilde{\nu}}{\partial t} + \nabla.(U\tilde{\nu}) = c_{b1}\bar{S}\,\tilde{\nu} + \frac{1}{\sigma}[\nabla.((\nu + \tilde{\nu})\nabla\tilde{\nu} + c_{b2}(\nabla\tilde{\nu})^2] - c_{w1}f_w(\frac{\tilde{\nu}}{d})^2 \qquad [3.2.51]$$

which makes numerical solutions easier and tends to be linear on solving. $\tilde{\nu}$ is identical to ν_t, far away from the wall.

$c_{b1}\bar{S}\,\tilde{\nu}$: This term is for turbulence generation in the model, even in the regions of high shear to replicate turbulence generation in real flows. This is done by assuming shear rate to be proportional to the mean velocity profile. 'S' is the shear rate tensor.

$$S' = \frac{1}{2}(\nabla U + (\nabla U)^T), \qquad \text{and, } S = \sqrt{S':S'} \qquad [3.2.52]$$

$\frac{1}{\sigma}[\nabla.((\nu + \tilde{\nu})\nabla\tilde{\nu} + c_{b2}(\nabla\tilde{\nu})^2]$: The diffusion term here has an additional non-linear term. This term is often split into a linear diffusion term, for the finite volume discretization and a non-linear explicit source term.

The term that shows the damping of turbulence near to any wall is, $c_{w1}f_w(\frac{\tilde{\nu}}{d})^2$. This is done by a combination of inviscid damping of pressure fluctuations and the viscous damping very close to the wall. The negative sign indicates destruction of turbulence. 'd' is the distance to the nearest wall and lesser 'd' indicates more turbulence destruction. The viscous damping is done by f_{v1} Eq 3.2.50 that imposes the quartic behavior close to the wall on nut which implicitly contains a lot of the viscous destruction terms and inviscid damping of pressure fluctuations is done by this term.

$f_w \rightarrow 0$ as $d \rightarrow 0$ to prevent division by zero error.

The boundary conditions for $\tilde{\nu}$ is taken as $\tilde{\nu} = 0$ at the wall for its linear behaviour in the viscous sub-layer. In the freestream, that is, far away from the wall, $\tilde{\nu}$ is taken to be the same as ν_t. So, at the inlet to the domain,

$$\tilde{\nu} = \nu_t = \frac{C_\mu k^2}{\epsilon} \text{ or } \qquad \tilde{\nu} = \nu_t = \frac{k}{\omega} \qquad [3.2.53]$$

k and ω can be calculated from turbulent length scale, typically 10% of the aerofoil chord length and turbulence intensity, I, typically of 5%,

$$k = \frac{3}{2}U_\infty^2 I^2 \qquad \epsilon = C_\mu \frac{k^{3/2}}{l} \qquad [3.2.54]$$

This term is important at the wake edge of the turbulent region, where diffusion dominates. Away from the boundary, if there is a wake region, there will be more spreading and more diffusion. Thus, to control this spreading and bring the numerical simulations close to that of the experiments, c_{b2} was calibrated to be 0.622 that gives a more accurate spreading of the wake profile. This is useful in the downstream of an aerofoil or a diffusion section in a duct.

6) Realizable $\mathbf{k-\varepsilon}$ model: [74]

In this model, a new model dissipation rate Equation and a new realizable eddy viscosity formulation, was proposed. The new model dissipation rate Equation was based on the dynamic Equation of the mean-square vorticity fluctuation at Reynolds number of large turbulent scale. The new eddy viscosity formulation is based on the realizability constraints: normal Reynolds stress positivity and Schwarz' inequality for turbulent shear stresses. The flows examined included:

(1) rotating homogeneous shear flows;
(2) boundary-free shear flows including a mixing layer, planar and round jets;
(3) a channel flow, and flat plate boundary layers with and without a pressure gradient; and
(4) backward facing step separated flows.

This model was proved to be significantly better than the standard k-epsilon eddy viscosity model.

Large Eddy Simulation

[*The equation number starts from 3.3.8 for this section*]

The calculation for the various parameters through Large Eddy Simulations as a continuation of Section 3.3 of Chapter-3 is detailed below.

Part-1: Calculation of Mean Velocity

The CFD code computes the instantaneous velocity (U). The mean velocity is calculated by time-averaging ($\bar{U}$). The averaging should be done after initial transient. Once the mean velocity field is computed by the RANS model, the fluctuating velocity component (u′) can conceptually be estimated by subtracting time-averaged mean velocity, $\bar{U}$ from instantaneous velocity, U. To isolate the fluctuations, u'

$$u' = U - \bar{U} \qquad [3.3.8]$$

There are 3 fluctuating velocity components, the other two being,

$$v' = V - \bar{V} \qquad [3.3.9]$$

$$w' = W - \bar{W} \qquad [3.3.10]$$

Part-2: Fluctuating Velocity

The kinetic energy is calculated in the fluctuations, which is resolved by the mesh. The reason for 'resolved' is; in each of the cells the fluctuating velocity component is calculated, thus resolved by cells, that is, mesh. But in RANS u' was modelled, not calculated. Thus, 'k' would not be resolved in RANS calculation. Kinetic energy per unit mass is $\frac{1}{2}u * u$, so the fluctuating velocity components are to be multiplied together.

The instantaneous Reynolds stresses that are resolved by the mesh are:

$$u'u', u'v', u'w', v'u', v'v', v'w', w'u', w'v', w'w' \quad [3.3.11]$$

The normal components are only used to calculate the resolved turbulent kinetic energy. Time average is calculated for each of the product of fluctuating velocity components.

$$\overline{u'u'}, \overline{u'v'}, \overline{u'w'}, \overline{v'u'}, \overline{v'v'}, \overline{v'w'}, \overline{w'u'}, \overline{w'v'}, \overline{w'w'} \quad [3.3.12]$$

The average of the fluctuating velocity is zero

$$\overline{u'} = 0 \quad [3.3.13]$$

This is the reason the time averaging is done after multiplication. Taking the time average first will lead to the value of zero as the area under the curve above x-axis and below it will be same. So, the fluctuating components are to be multiplied first and then time averaged.

Part-3: Resolved turbulent kinetic energy

There are 9 components of the resolved Reynolds Stress tensor per unit density:

$$\frac{R_{ij}}{\rho} = \begin{bmatrix} \overline{u'u'} & \overline{u'v'} & \overline{u'w'} \\ \overline{v'u'} & \overline{v'v'} & \overline{v'w'} \\ \overline{w'u'} & \overline{w'v'} & \overline{w'w'} \end{bmatrix} \quad [3.3.14]$$

The order of multiplication does not matter, by the commutative law, thus

$$\overline{u'v'} = \overline{v'u'}, \quad \overline{u'w'} = \overline{w'u'}, \ \overline{v'w'} = \overline{w'v'} \quad [3.3.15]$$

Therefore, there are only 6 independent quantities (symmetric tensor)

$$\frac{R_{ij}}{\rho} = \begin{bmatrix} \overline{u'u'} & \overline{u'v'} & \overline{u'w'} \\ & \overline{v'v'} & \overline{v'w'} \\ & & \overline{w'w'} \end{bmatrix} \quad [3.3.16]$$

The diagonal components are used to calculate the resolved turbulent kinetic energy

$$k_{res} = \frac{1}{2}(\overline{u'u'} + \overline{v'v'} + \overline{w'w'}) \quad [3.3.17]$$

It is also written as

$$k_{res} = \frac{1}{2}\left(\overline{u'^2} + \overline{v'^2} + \overline{w'^2}\right) \quad [3.3.18]$$

The reason of adding the diagonal components is that within the flow field, there are some turbulence in the x,y and z-axis, but the turbulent kinetic energy at a point is of concern, which is a scalar value. In OpenFOAM, the matrix calculation (units: m^2/s^2) of Eq 3.3.16 is done with the command uPrime2Mean. ANSYS Fluent calculates the root mean square velocity (units: m/s)

$$U_{RMSE} = \sqrt{\overline{(u')^2}} \quad [3.3.19]$$

$$k_{res} = \frac{1}{2}\left({U_{RMSE}}^2 + {V_{RMSE}}^2 + {W_{RMSE}}^2\right) \quad [3.3.20]$$

To calculate k_{res} a new field is to be calculated in the post-processor.

$$k_{res} = 0.5 * (\text{UPrime2MeanXX} + \text{UPrime2MeanYY} + \text{UPrime2MeanZZ}) \quad [3.3.21]$$

$$k_{res} = 0.5 * (\text{URMSE} * \text{URMSE} + \text{VRMSE} * \text{VRMSE} + \text{WRMSE} * \text{WRMSE}) \quad [3.3.22]$$

Part-4: Total Turbulent Kinetic Energy

k_{res} is the turbulent kinetic energy resolved by the mesh, not the total kinetic energy (k).

$$k = k_{res} + k_{sgs} \quad [3.3.23]$$

In RANS, 'k' is modelled and in LES 'k' is calculated via k_{res} and k_{sgs}.

Part-5: Sub-Grid Scale Turbulent Kinetic Energy

k_{sgs} is the turbulent kinetic energy of the eddies smaller than mesh size. OpenFOAM calculates k_{sgs} for the users and writes it as a field (k). But, ANSYS Fluent users need to calculate k_{sgs} in the post-processor. The method used to determine k_{sgs} depends on the sub-grid scale (SGS) model. The CFD codes solve a transport equation for k_{sgs} if a kinetic energy transport model is used,

$$\frac{\partial(\rho k_{sgs})}{\partial t} + \nabla.\left(\rho U k_{sgs}\right) = \nabla.\left(\rho D_k \nabla k_{sgs}\right) - C_{\in}\frac{\rho k_{sgs}^{3/2}}{\Delta} + \rho G_{sgs} - \frac{2}{3}\rho k_{sgs}\nabla.U \quad [3.3.24]$$

The post-processor then directly reads k_{sgs}. But for other models, k_{sgs} is to be calculated. It can be calculated from the sub-grid length scale, l_{sgs}.

$$k_{sgs} = \left(\frac{\mu_{sgs}}{\rho l_{sgs}}\right)^2 \quad [3.3.25]$$

Part-6: Sub-grid Length scale

l_{sgs} can be described as the size of an eddy that has same turbulent kinetic energy as the average of all the eddies smaller than the mesh size, similar to the integral length scale (l_0). l_{sgs} should be smaller than the mesh size,

$$l_{sgs} = C_s * (Cell\ Volume)^{1/3} \quad [3.3.26]$$

C_s is the Smagorinsky coefficient which is equal to 0.1. C_s is less than 1 in order to have the length scale smaller than the cell size. The eddies are damped near the walls. l_{sgs} is limited by the distance to the wall, y, for thin high aspect ratio cells. So, l_{sgs} can be expressed as

$$l_{sgs} = \min(\kappa y, C_s \Delta^{1/3}), \qquad \kappa = 0.41 \quad [3.3.27]$$

Part-7: Turbulent Kinetic Energy

After obtaining both the residual turbulent kinetic energy, k_{res} and sub-grid scale turbulent kinetic energy, k_{sgs}, the ratio $\frac{k_{res}}{(k_{res}+k_{sgs})}$ is calculated in the post-processor to check whether the mesh is resolving more than 80% of the turbulent kinetic energy.

Part-8: Mesh Refinement

Mesh refinement increases k_{res} and decreases k_{sgs}. It leads to change in balance of the two types of turbulent kinetic energy.

Understanding the sub-grid models:

Large turbulent eddies are unstable in nature and tend to break down into smaller and even smaller eddies, by a process known as eddy or energy cascade. This cascading action leads to smallest eddies that dissipates into heat through molecular viscosity. The smallest eddies just larger than the cell size, that on further breakdown creates eddies which cannot be resolved, and are also not small enough to be dissipated to heat by molecular viscosity, are removed by increasing the turbulent kinetic energy.

Turbulence Dissipation Rate (ϵ) is the rate at which turbulent kinetic energy is converted into thermal energy, that is, turbulent kinetic energy per unit time (SI unit: m^2/s^2). An increase in Turbulence dissipation rate (ϵ) leads to an increase in the rate at which the turbulent eddies are dissipated. It is calculated by the product of molecular viscosity and two velocity gradients.

$$\epsilon = \nu \frac{\partial U_i}{\partial x_j} \frac{\partial U_i}{\partial x_j} \quad [3.3.28]$$

The velocity gradient considers the mean and fluctuating components of velocity. As the fluctuations are neglected in RANS, so a transport equation for Turbulence Dissipation Rate (ϵ) is solved. But in comparison to the energy of the flow, the molecular viscosity is very small. As the eddies get smaller in size, the velocity gradient increases till the molecular viscosity are large enough to dissipate them. To dissipate these eddies, the viscosity is increased from ν to $\nu + \nu_{sgs}$, where ν_{sgs} is the sub-grid scale viscosity. Due to this, the dissipation rate also increases as,

$$\epsilon = (\nu + \nu_{sgs}) \frac{\partial U_i}{\partial x_j} \frac{\partial U_i}{\partial x_j} \quad [3.3.29]$$

But ν is increased specifically to remove the targeted eddies just larger than the cell size. Thus, this is not the true dissipation rate, rather this artificial dissipation mimics the breakdown process.

Increasing dissipation in LES

Starting with the Navier Stokes and continuity equation,

$$\frac{\partial \rho}{\partial t} + \frac{\partial (\rho U_j)}{\partial x_j} = 0 \quad [3.3.30]$$

$$\frac{\partial \rho U_j}{\partial t} + \frac{\partial (\rho U_i U_j)}{\partial x_j} = -\frac{\partial P}{\partial x_j} + \frac{\partial}{\partial x_j}(\tau_{ij}) \quad [3.3.31]$$

τ_{ij} is the viscous stress term, that dissipates energy through molecular viscosity. Its strength is only to dissipate the smallest (Kolmogorov) eddies and not the ones just larger than the grid size. So, to dissipate these eddies just larger than grid size, the dissipation is increased by adding an extra stress term, τ_{sgs} and Eq 3.3.31 becomes,

$$\frac{\partial \rho U_j}{\partial t} + \frac{\partial (\rho U_i U_j)}{\partial x_j} = -\frac{\partial P}{\partial x_j} + \frac{\partial}{\partial x_j}(\tau_{ij} + \tau_{sgs}) \quad [32]$$

This term can be mathematically derived by filtering the Navier-Stokes equations. The eddies smaller than the mesh size are not directly resolved but act as a resistive force on the eddies that are resolved. So instead of trying to resolve those small eddies, the resistive force is modelled. This resistive force dissipates energy and breaks down the eddies just larger than the cell size.

The sub-grid stress can be calculated with an eddy-viscosity model. As the eddies get smaller, the velocity gradient increases, the sub-grid stress and force also increases. This sub-grid force is increased enough so that the eddies just larger than cell size is dissipated.

$$\tau_{sgs} = 2\rho\nu_{sgs}S_{ij}^{*} - \frac{2}{3}\rho k_{sgs}\delta_{ij} \quad [3.3.33]$$

$$where\ S_{ij}^{*} = \frac{1}{2}\left(\frac{\partial\widetilde{U}_i}{\partial x_j} + \frac{\partial\widetilde{U}_j}{\partial x_j} - \frac{1}{3}\frac{\partial\widetilde{U}_k}{\partial x_k}\delta_{ij}\right) \quad [3.3.33a]$$

S_{ij}^{*} is the strain rate of the resolved eddies on the CFD mesh.

ν_{sgs} is the factor that controls the strength of the stress that is applied. If ν_{sgs} is too high, the eddies will break down and the kinetic energy will drop too low. But if there is no ν_{sgs} then the eddies just larger than cell size won't get dissipated as τ_{sgs} will not be large enough. So, ν_{sgs} is the controller of the strength of the stress τ_{sgs}. The calculation of ν_{sgs} is described next.

It is assumed that the eddies smaller than the cell size are isotropic, as shown in the figure. Isotropic means the eddies have the same shape but not the same size. Thus, the eddy size is essential to be considered and not the eddy shape. As the eddies are isotropic, the value of sub-grid scale viscosity, ν_{sgs} will be scalar, which depend on the size of the eddies. Larger cells will have larger eddies and will require higher ν_{sgs}. There are various sub-grid models to calculate ν_{sgs}, but all of those are based on the same principle of the eddies to be isotropic. The ν_{sgs} required on a coarse mesh will be larger than that for a fine mesh. Thus, the sub-grid viscosity can be expressed as a function of the mesh size,

$$\nu_{sgs} = f(\Delta) \quad [3.3.34]$$

This means that different equations are solved on each mesh with different sub-grid viscosity. So, a mesh independence study cannot be done for LES, instead the criteria of resolving 80% of the turbulent kinetic energy is considered to be a remarkable LES. On the other hand, in RANS calculation, the equations solved for eddy viscosity is same throughout even if the resolution of the field changes.

The Smagorinsky Turbulence Model

With the assumption of sub-grid scale eddies being isotropic, the determination of ν_{sgs} is carried out. According to Smagorinsky, the kinematic viscosity is decomposed into a product of velocity and length from the perspective of units.

$$Kinematic\ viscosity\ (m^2/s) = Velocity\ (m/s) \times Length\ (m)$$

$$\text{So, } \nu_{sgs} \sim U_0 \times l_0 \qquad [3.3.35]$$

As the eddies are isotropic, so a length scale (l_0) is enough to categorize the shape of the eddies. The average velocity in Cartesian coordinates is zero. So, a velocity that characterizes the eddy is chosen. It is better if the velocity difference (ΔU) across the eddy is known. It can be calculated as:

$$\Delta U = l_0 * \frac{\partial U}{\partial z} \qquad [3.3.36]$$

Thus, the velocity scale is

$$U_0 \sim l_0 * \frac{\partial U}{\partial z} \qquad [3.3.37]$$

Now, if instead of a vertical line through an eddy, a horizontal line is drawn, as shown in Figure, the velocity scale (Eq 3.3.37) shall be

$$U_0 \sim l_0 * \frac{\partial U}{\partial z} \quad or, \; U_0 \sim l_0 * \frac{\partial V}{\partial y} \qquad [3.3.38]$$

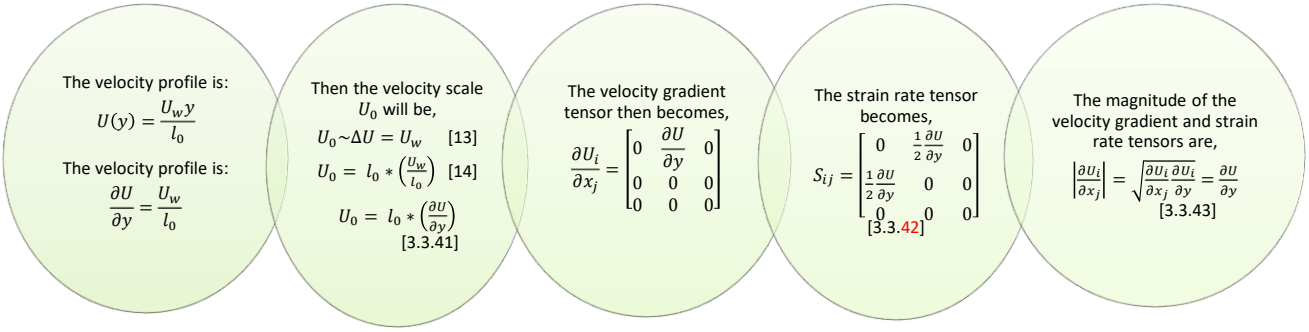
To get a more generalized form of velocity scale for any coordinate system in any direction, the method used in Smagorinsky model is the method of strain rate tensor (S_{ij}).

$$S_{ij} = \frac{1}{2}\left(\frac{\partial U_i}{\partial x_j} + \frac{\partial U_j}{\partial x_i}\right) \qquad [3.3.39]$$

$$S_{ij} = \begin{bmatrix} \frac{\partial U}{\partial x} & \frac{1}{2}\left(\frac{\partial U}{\partial y} + \frac{\partial V}{\partial x}\right) & \frac{1}{2}\left(\frac{\partial U}{\partial z} + \frac{\partial W}{\partial x}\right) \\ \frac{1}{2}\left(\frac{\partial V}{\partial x} + \frac{\partial U}{\partial y}\right) & \frac{\partial V}{\partial y} & \frac{1}{2}\left(\frac{\partial V}{\partial z} + \frac{\partial W}{\partial y}\right) \\ \frac{1}{2}\left(\frac{\partial W}{\partial x} + \frac{\partial U}{\partial z}\right) & \frac{1}{2}\left(\frac{\partial W}{\partial y} + \frac{\partial V}{\partial z}\right) & \frac{\partial W}{\partial z} \end{bmatrix} \qquad [3.3.40]$$

But a scalar value is required for the velocity scale. The magnitude of S_{ij} would be an incorrect approach. To find S_{ij}, an example can be considered,

A shear flow driven by a moving wall U_w is considered. The upper wall is moving with a velocity of U_w and the bottom wall is stationary. The distance between the two walls is l_0 and the axis system is followed as shown.



Thus, the magnitude of strain rate tensor (S_{ij}) comes as,

$$\left|S_{ij}\right| = \sqrt{S_{ij}S_{ij}}\sqrt{\left(\frac{1}{2}\frac{\partial U}{\partial y}\right)^2 + \left(\frac{1}{2}\frac{\partial U}{\partial y}\right)^2} = \frac{1}{\sqrt{2}}\frac{\partial U}{\partial y} \quad [3.3.44]$$

Hence, if the strain rate tensor is to be used for the velocity scale U_0, $\sqrt{2}$ is to multiplied,

$$U_0 \sim l_0 * \sqrt{2S_{ij}S_{ij}} \quad [3.3.45]$$

So, the expression for the sub-grid scale kinematic viscosity (ν_{sgs}) becomes,

$$\nu_{\text{sgs}} = U_0 \times l_0 \quad [3.3.46]$$

The velocity scale is substituted as,

$$\nu_{\text{sgs}} = \left(l_0 \times \sqrt{2S_{ij}S_{ij}}\right) \times l_0 \quad [3.3.47]$$

$$\nu_{\text{sgs}} = {l_0}^2\sqrt{2S_{ij}S_{ij}} \quad [3.3.48]$$

Therefore, the strain rate tensor can be calculated from the smallest resolved eddies. Next, the calculation of characteristic length scale, l_0 is required. The size of the eddies is calculated as they are assumed to be isotropic. The sub-grid scale eddies are actually a range of different sized eddies. So, l_0 is chosen such that, the eddy contains the same turbulent kinetic energy as that of an average sized eddy, smaller than the cell size. This is conceptually similar to that of integral length scale. l_0 is expected to be smaller than the cell size and thus can be expressed as,

$$l_0 = C_s\Delta, \text{ where } 0 < C_s < 1 \quad [3.3.49]$$

C_s is the Smagorinsky coefficient and can be called as a fraction of the cell size that calculates the sub-grid length scale. Substituting Eq 3.3.49 in Eq 3.3.48,

$$\nu_{\text{sgs}} = (C_s \Delta)^2 \sqrt{2 S_{ij} S_{ij}} \quad [3.3.50]$$

The value of Smagorinksy Coefficient, C_s for homogenous isotropic turbulence (HIT) was derived by [89]. HIT means the turbulence was assumed to be far away from walls, that is, with no shear. [89] referred the explanation by [90]for the derivation of C_s,

$$C_s = \frac{1}{\pi} \left(\frac{2}{3 C_\kappa} \right)^{3/4} \quad [3.3.51]$$

C_k is called Kolmogorov constant, derived from Kolmogorov energy spectrum and found out to be 1.5.

$$E(\kappa) = C_\kappa \epsilon^{2/3} \kappa^{-5/3} \quad [3.3.52]$$

So, for homogenous isotropic turbulence, after solving analytically, $C_s = 0.173$. But at regions of high shear or near the walls, this value of C_s leads to high dissipation of the eddies. In modern CFD codes the value of C_s is chosen as follows,

ANSYS Fluent: $C_s = 0.1$; Phoenics: $C_s = 0.17$

The value of C_s should be checked and noted before running simulations.

The limitation of the original model is that a variable sub-grid length, l_0 and C_s is required as the sub-grid eddies vary throughout the domain, particularly near the walls. There is a region of laminar flow in the viscous sub-layer (y+<5), close to the walls. Here, the sub-grid scale eddies are small enough to get dissipated by the viscosity. Hence, there shouldn't be any sub-grid eddies here. Some damping of eddies is expected at the buffer and log-law region.

In the original Smagorinsky model [91], the sub-grid eddies are represented as an additional stress term, τ_{sgs}. This mimics the energy cascade and breaks down the eddies just larger than grid size. Absence of sub-grid eddies mean $\tau_{sgs} = 0$. In Eq 3.3.33, the second term ($\frac{2}{3} \rho k_{sgs} \delta_{ij}$) is often neglected by merging it with resolved pressure to obtain a modified pressure. Now, it is to be checked whether the τ_{sgs} reaches zero towards the wall, in the original Smagorinsky model [91]. In the laminar flow region, the velocity profile is linear and the strain rate tensor is as given in Eq 3.3.42.

Thus,$S_{ij}^* \neq 0$ and from that,

$$\tau_{sgs} = 2 \rho \nu_{sgs} S_{ij}^* \quad [3.3.53]$$

$\tau_{sgs} \neq 0, unless\ \nu_{sgs} = 0$, but in the original Smagorinsky model [91], C_s~0.17 and the cell size $\Delta > 0$, so from Eq 3.3.50 it is understood that $\nu_{sgs} \neq 0$ even in the viscous sub-layer. Thus, the original Smagorinsky model [91] does not think that the sub-grid eddies near the wall are dissipated, as it happens practically. These sub-grid eddies are to be damped. The different options for correcting ν_{sgs} near the wall are,

- SGS Kinetic Energy: Model is to be changed
- Van Driest Damping Dynamic Smagorinsky: Length scale is to be changed
- WALE: Velocity scale is to be changed

The Van Driest Damping:

The eddies tend to get smaller as they approach the walls. This is because there are two wall conditions to be satisfied,

i) No penetration condition: The fluid does not penetrate the wall

ii) No slip condition: The tangential velocity of the fluid close to the wall is zero or of the same velocity as that of the wall.

These conditions apply to both mean flow and fluctuations, that is, the eddies.

The rotational flow from the eddies is blocked by the wall, thus there should not be any eddy larger than the distance to the wall. But eddy of any size tends to move and when it reaches the wall, it gets blocked. This has a net effect of reducing the size of the eddies close to the wall. Hence, the eddies smaller than the distance to the wall are also damped by the wall.

The eddy size close to the wall is quantifies using a RANS approach, by solving the RANS equations close to the wall, using an eddy-viscosity approach to calculate the Reynolds stresses,

$$-\rho\overline{u'v'} = \rho\upsilon_T \frac{\partial u}{\partial y} \qquad [3.3.54]$$

This Eq 3.3.54 is used by [90] to derive a length scale for all the eddies close to the wall. In the logarithmic region, the mean velocity profile is modelled by,

$$U^+ = \frac{1}{\kappa}\log(y^+) + C \qquad [3.3.55]$$

Therefore, the mean velocity gradient is,

$$\frac{\partial U^+}{\partial y^+} = \frac{1}{\kappa y^+} \qquad \text{Or, } \frac{\partial U}{\partial y} = \frac{u_\tau}{\kappa y} \qquad [3.3.56]$$

Since, $y^+ = \frac{\rho u_T y_p}{\mu}$

In the logarithmic or log-law region the Reynolds shear stress is relatively constant and balances the wall shear stress as,

$$-\rho\overline{u'v'} = \tau_w = \rho u_\tau^2 \quad [3.3.57]$$

Substituting Eq 3.3.57 and Eq 3.3.56 in Eddy viscosity equation,

$$-\rho\overline{u'v'} = \rho\nu_T \frac{\partial U}{\partial y} \quad [3.3.58]$$

$$\rho u_T^2 = \rho\nu_T \left(\frac{u_\tau}{\kappa y}\right) \quad [3.3.59]$$

$$\nu_T = u_T \kappa y \quad [3.3.60]$$

Kinematic viscosity being a product of length scale (mixing length, l_m) and velocity scale (square root of Reynolds shear stress), and from Eq 3.3.60,

$$u_T \kappa y = l_m * \left(\overline{u'v'}\right)^{1/2} \quad [3.3.61]$$

Solving for mixing length from Eq 3.3.61,

$$u_\tau \kappa y = l_m * (u_\tau^2)^{1/2} \quad [3.3.62]$$

$$l_m = \kappa y \quad [3.3.64]$$

The mixing length can be called a measure of how large the eddies are. This is a metric of the eddy size in the logarithmic region. From Eq 3.3.64, which is a logarithmic solution, it is understood that as y reduces or tends to zero, the mixing length also tends to zero, that is, eddies get smaller on approaching the wall.

Concept of mixing length

Considering that a fluid volume being transported by large eddies which dissipate into smaller eddies, mixing length can be defined as the distance travelled by this fluid volume prior to its dissipation into smaller eddies. It is not similar to the integral length scale, rather smaller than that but it is larger than the sub-grid length, because it is being dissipated into smaller eddies. So, it can be written as,

$$l_0 = \min(l_m, C_s\Delta) \quad [3.3.65]$$

Substituting Eq 3.3.64 in Eq 3.3.65,

$$l_0 = \min(\kappa y, C_s \Delta) \quad [3.3.66]$$

Eq 3.3.66 will reduce the sub-grid length scale in the logarithmic region.

For the viscous sub-layer and buffer region, it is intended to get a continuous solution to the logarithmic region. The Van-Driest proposed a solution for this, mentioned in [90](1986). The coefficient A+ = 26 makes the velocity profile have good agreement with experimental data.

$$U^+ = \int_0^{y^+} \frac{2\,dy'}{1+\left(1+4\left(l_m^+\right)^2\right)^{1/2}} \quad [67]$$

$$\text{Where, } l_m = \kappa y \left(1 - \exp\left(-\frac{y^+}{A^+}\right)\right) \quad [3.3.68]$$

Eq 3.3.68 is written as $l_m = \kappa y * D$, where $D = \left(1 - \exp\left(-\frac{y^+}{A^+}\right)\right)$ and is known as Van-Driest Damping Function. D tends to zero close to the wall and tends to 1 far away from the wall, in the logarithmic region where viscous effects are small. Thus, D can be expressed as a function which accounts for viscous effects. Putting Eq 3.3.68 in Eq 3.3.65,

$$l_0 = \min\left[\kappa y * \left(1 - \exp\left(\frac{y^+}{A^+}\right)\right), C_s \Delta\right] \quad [3.3.69]$$

This is the Van-Driest Dynamic Smagorinksy model. This mathematical form is used differently by different CFD codes.

ANNEXURE-II: NUMERICAL METHODOLOGY

Annexure-II: Section-4.3

The detailed numerical methodology used in the problem stated in Section-5.3 is given here.

Annexure-II: Section-4.3.2

DynamicMeshDict

Diffusivity Parameter: This is an optional parameter in OpenFOAM which provide default values if not included. It controls how the mesh motion is distributed through the mesh. It is assumed that there is a moving boundary and a set of static boundaries. The mesh motion solver diffuses the motion of the structure boundary into the domain. The list of diffusivity options are:

inverseDistance: To reduce mesh morphing inverse to distance from a series of patches. Less mesh morphing, far away from the specified patches. This applies in every directions from the structure.

- inverseFaceDistance
- inversePointDistance
- inverseVolume
- uniform

Forces Definitions: The restraints act as reactionary forces other than the fluid interaction. Fluid interaction is automatically included whenever a patch is specified as part of the body. The restraint forces model other forces through simple reaction force terms. The exact force magnitude will change depending on body behavior. There can be multiple force types, that is, restraint conditions in this library.

In addition to the fluid interaction, the constraints function as reactive forces. Every time a patch is designated as a component of the body, fluid interaction is automatically taken into account. Through straightforward reaction force concepts, the restraint forces represent other forces. Depending on how the body responds, the precise force magnitude will differ. Multiple force kinds, or constraint conditions, can exist in this library.

Force type keywords available in OpenFOAM are:

i) constantForce: The constantForce is a constant force that is unaffected by the force vector-specified body motions. This specifies the force's magnitude and direction. With body motions, the direction of force does not rotate but remain fixed. An attachment point is also necessary per the definition. If the constantForce is not attached at the same coordinates as the centre of gravity, moment coupling will be taken into account by the sixDoF solver. With bodily motions, the reference attachment point will also move. The relative position to the body's centre of gravity will remain unchanged. To account for any unmodeled force elements and balance out the concerned model forces, the constantForce term is particularly helpful.

ii) linearDamper: The straight reactive force that is proportionate to the body velocity is provided by the damper force. The definition of a basic damping coefficient. Only linear motions are responsive to the linear damping force. The centre of gravity of the body receives all damper forces. The damping force is consistently provided in a direction that is equidistant from the body's vector of linear motion.

iii) sphericalAngularDamper: The reactive force delivered by the sphericalAngularDamper is related to the body's angular velocity. The definition of a basic damping coefficient. Only angular motions are responsive to the angular damping force. The centre of gravity of the body receives

all damper forces. Every time the damping force is supplied, it is done so in the exact opposite direction of the body's angular velocity vector.

iv) linearSpring: This section is explained in Section-4.3.2

v) linearAxialAngularSpring: A more complete model for restraint forces that are precisely targeted at body rotational motion is provided by the linearAxialAngularSpring. Moments are produced by the force model that revolve around the specified axis. Moments are always reactive and run anticlockwise to how the body is rotating. The force in this model is only dependent on rotation about the designated axis. The following are the main characteristics of this model stiffness, force proportional to body rotation.

- axis, specifies the axis of rotation: The rotation axis specifies the rotation's reference axis. The rotation of the designated axis serves as the foundation for all forces in this model. A vector is used to specify the axis. Whatever vector is defined, OpenFOAM will automatically extract the unit vector. At the body's centre of gravity, everything happens.

- damping, force proportional to body angular velocity: The linearAxialAngularSpring moments' spring component is represented by this stiffness. This identifies the moment's component that reacts to the angle of body rotation. The rotation angle with respect to the referenceOrientation is the basis for the moments. The units of stiffness are Nm/rad. This reactive phase can alternatively be defined as having zero rigidity. This will result in an object that only functions as a damper. The linearAxialAngularSpring moments' damper component is this damping. This describes the moment's component that reacts to the body's angular motion. Based on the angular velocity at the body centroid, the moments are calculated. The unit of measurement for damping is Nms/rad. This reactionary force moment can also be defined with a dampening value of 0. That will result in a thing that just acts as a spring.

- referenceOrientation, allows to provide initial conditions of rotational moment for the spring: The force model's neutral moment orientation is specified by the reference orientation. A vector is used to specify orientation. The rotation of the body orientation in radians with respect to the reference orientation chosen will be calculated using the force model. Only specified rotations that are about the axis of rotation, are taken into account by the model.

vi) tabluatedAxialAngularSpring

Motion definitions: A set of restricted motions is referred to as constraints. The permissible movements of the body are directly controlled by constraints. These don't apply reactive forces like restraints do. There is no force created by constraints. They only limit the body's movements. A collection of established constraints makes up the constraints parameters. To limit body motions in various ways, one can construct combinations of different constraint kinds.

List of MotionType keywords

i) axis: Only rotational motion is constrained by the axis constraint. Motion is linear and unrestrained. Only rotation about the specified axis of rotation is permitted by the axis limitation. This restricts the range of possible body motions to one degree of freedom for rotation and three degrees for linear motion. Based on the orientation of a vector, the axis is defined. The axis definition will be automatically scaled by OpenFOAM such that it is seen as a unit vector definition.

ii) line

iii) plane

iv) point: The body is given unrestricted rotational motion about the specified centre of rotation, but the point constraint eliminates all linear motion. This enables 360-degree rotation. The "centreOfR" keyword specifies the rotational centre. The centre of gravity of the body need not be here. With the body rotating around the joint, this can be used to define a spherical joint. Multiple constraint definitions can be added. A pinned connection would be replicated by a point constraint and an axis constraint.

v) orientation

Solver Type: This option is present in dynamicMeshDict. The types of solvers that any OpenFOAM user can opt are explained next.

Newmark – β Method

By using numerical space and temporal discretization, the response of a damped or undamped structural system can be assessed using this method. Any kind of loading can be done using this technique. The only prerequisite is that it is possible to identify the time series of the external excitation. The generic stimulated structural system's second-order differential equation can be expressed as,

$$\mathbf{M\ddot{U} + C\dot{U} + KU = F_{ext}} \qquad \textbf{[4.4.1.1]}$$

Using extended mean value theorem, the first time-derivative of the motion is expressed as

$$\dot{U}_{n+1} = \dot{U}_n + \Delta t \ddot{U}_\alpha \qquad [4.4.1.2]$$

Where, $\ddot{U}_\gamma = (1-\alpha)\ddot{U}_n + \alpha\ddot{U}_{n+1}, 0 \leq \alpha \leq 1$

This transforms the Eq. 4.4.1.2 into

$$\dot{U}_{n+1} = \dot{U}_n + (1-\alpha)\Delta t\ddot{U}_n + \alpha\Delta t\ddot{U}_{n+1} \quad [4.4.1.3]$$

As to incorporate the temporal change of the second time derivative of the motion, in the estimation of the displacement, the following formulations are suggested,

$$U_{n+1} = U_n + \Delta t\dot{U}_n + \frac{1}{2}\Delta t^2\ddot{U}_\beta \quad [4.4.1.4]$$

Where, $\ddot{U}_\beta = (1-2\beta)\ddot{U}_n + 2\beta\ddot{U}_{n+1}, 0 \le 2\beta \le 1$

Therefore, Eq. 4.4.1.4 becomes

$$U_{n+1} = U_n + \Delta t\dot{U}_n + \frac{1}{2}(1-2\beta)\Delta t^2\ddot{U}_n + \beta\Delta t^2\ddot{U}_{n+1} \quad [4.4.1.5]$$

The finite difference formulas for the Newmark-β scheme are

$$\ddot{U}_{n+1} = \frac{1}{\beta\Delta t^2}(U_{n+1} - U_n) - \frac{1}{\beta\Delta t}\dot{U}_n - \left(\frac{1}{2\beta} - 1\right)\ddot{U}_n \quad [4.4.1.6]$$

$$\dot{U}_{n+1} = \frac{1}{\beta\Delta t}(U_{n+1} - U_n) - \left(\frac{\alpha}{\beta} - 1\right)\dot{U}_n - \Delta t\left(\frac{\alpha}{2\beta} - 1\right)\ddot{U}_n \quad [4.4.1.7]$$

Crank-Nikolson Method

The Crank-Nicolson method is a numerical approach for solving partial differential equations (PDEs), especially parabolic equations. The outcome is a more precise and reliable numerical solution thanks to this implicit method that combines the forward and backward Euler methods. The second-order accuracy in time and aptitude for stiff equations of the Crank-Nicolson method make it a regularly utilized technique. Let's look at a standard parabolic equation with a single spatial dimension and a single instant of time:

$$\frac{\partial u}{\partial t} = \alpha\frac{\partial^2 u}{\partial x^2} = f(x,t) \quad [4.4.2.8]$$

where, the unknown function is denoted by $u(x,t)$, the diffusion coefficient is denoted by, and the source term is denoted by $f(x,t)$.

We discretize the temporal and spatial domains before applying the Crank-Nicolson approach. Assume that the grid is uniform and that the x- and t-axes are spaced at x and t, respectively. The following is a representation of the grid points for both space and time:

$x_i = i\Delta x$, where $i = 0, 1, 2, \ldots, N$

$t^n = n\Delta t$, where $n = 0, 1, 2, \ldots, M$

where N is the number of spatial grid points and M is the number of temporal grid points. Central differences will now be used to approximate the derivatives. We have the following for the second-order spatial derivative:

$$\frac{\partial^2 u}{\partial x^2} \approx \frac{U_{i+1}^n - 2U_i^n - U_{i-1}^n}{(\Delta x)^2} \qquad [4.4.2.9]$$

We may utilize the central difference to represent the first-order temporal derivative:

$$\frac{\partial u}{\partial t} \approx \frac{U_i^{n+1} - U_i^{n-1}}{(2\Delta t)} \qquad [4.4.2.10]$$

In the original PDE, we substitute these estimates, and we get:

$$\frac{U_i^{n+1} - U_i^{n-1}}{(\Delta t)} = \alpha \frac{U_{i+1}^n - 2U_i^n - U_{i-1}^n}{(\Delta x)^2} + f_i^n \qquad [4.4.2.11]$$

Introducing some constants to the notation to make it simpler:

$$\lambda = \alpha \frac{\Delta t}{(\Delta x)^2}$$

$$d = 2 + 2\lambda$$

With the equation now rearranged, we have:

$$-\lambda\, U_{i+1}^n + (d + 2\lambda)\, U_i^n - \lambda U_{i+1}^{n-1} = (d - 2\lambda) U_i^{n-1} + \lambda U_{i-1}^{n-1} + 2\Delta t f_i^n \qquad [4.4.2.12]$$

The unknown values of U at the (i, n+1) grid point are related to the values at the nearby grid points and the preceding time step using this equation. For all grid points in space ($i = 1, 2, \ldots, N-1$) and time steps ($n = 0, 1, \ldots, M-1$) this equation can be written.

These equations can be arranged as matrices as follows:

$$A\, U^{n+1} = B\, U^n + b^n \qquad [4.4.2.13]$$

where A is a $(N-1) \times (N-1)$ tridiagonal matrix with diagonal elements $(d + 2\lambda)$ and off-diagonal elements -λ, U^{n+1} is a column vector of unknowns at time step $(n+1)$, B is a $(N-1) \times (N-1)$ tridiagonal matrix with diagonal elements (d - 2λ) and off-diagonal elements λ, U^n is a column vector of known values at time step n, b^n is a column vector containing the source term and boundary conditions.

We have a choice of iterative methods, such as the conjugate gradient method, or direct methods, such as LU decomposition, to solve for U^{n+1}.

We may use the Crank-Nicolson method to iterate this procedure for each of the time steps ($n = 0, 1, \ldots, M-1$) to get the parabolic equation's numerical solution. In order to approximate parabolic PDEs with second-order accuracy in time, the Crank-Nicolson approach is a reliable and accurate method.

Symplectic Method

In order to solve ordinary differential equations (ODEs) with a symplectic structure, symplectic methods are numerical integration techniques that are frequently utilized. These techniques maintain the underlying system's symplectic geometry, which is useful for issues involving Hamiltonian mechanics.

The Störmer-Verlet method, often known as the leapfrog method, is one of the most well-known symplectic techniques. For Hamiltonian systems, it is a second-order technique that is frequently employed. The Störmer-Verlet method equations can be expressed as follows:

$x(t + h/2) = x(t) + (h/2) * v(t)$ is a position update.

Update on velocity: $F(x(t + \frac{h}{2})) = v(t) + (\frac{h}{2}) * v(t)$ [4.4.3.1]

Updated position: $x(t + h) = x(t + \frac{h}{2}) + (\frac{h}{2}) * v(t + \frac{h}{2})$ [4.4.3.2]

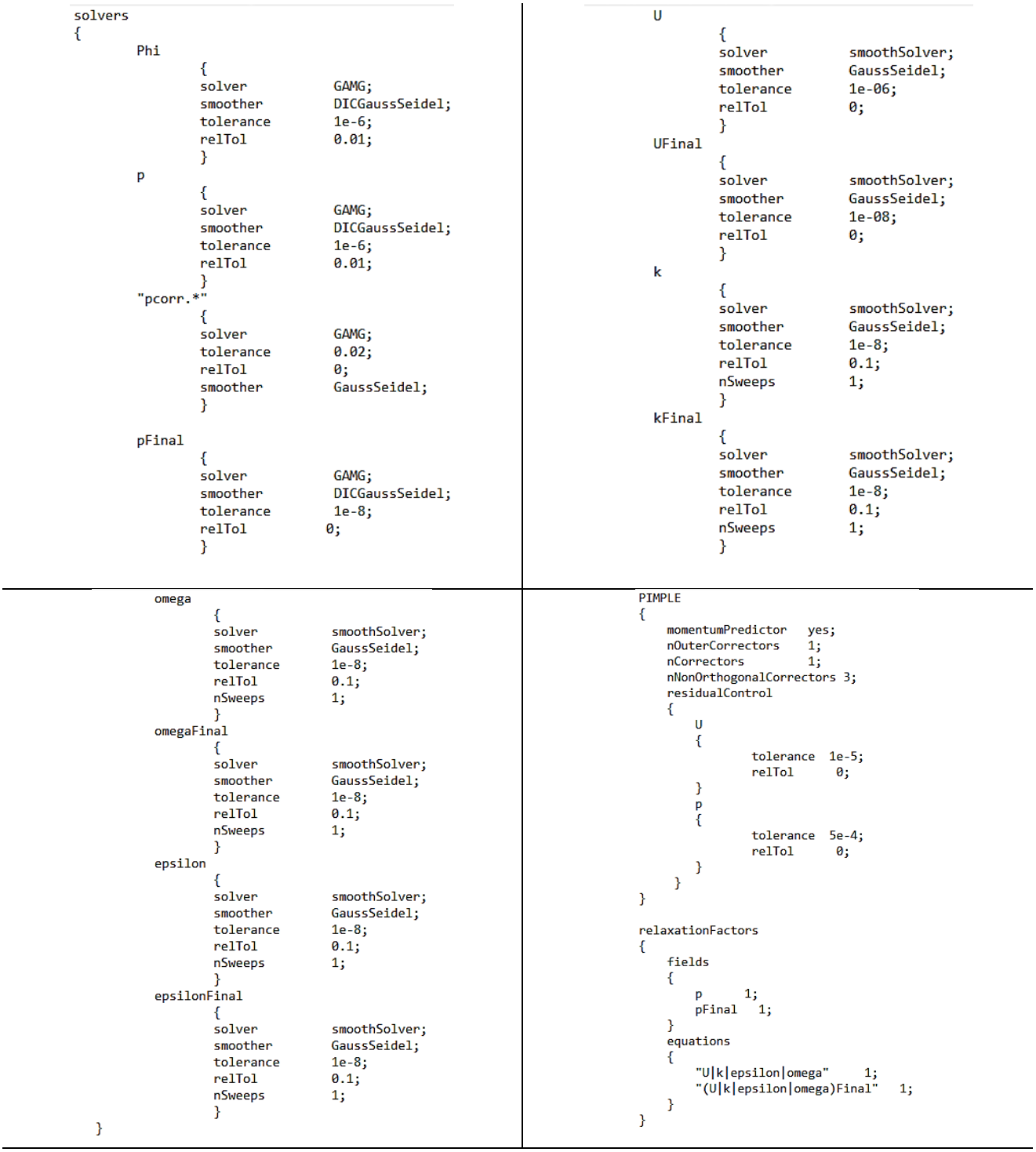
In this case, $x(t)$ denotes the system's position at time $t, v(t)$ denotes its velocity at that same time, $F(x(t))$ denotes the force acting on the system at that same position, h denotes the time step, and $t + h/2$ denotes the point in the middle of t and $t + h$.

These equations may not be relevant for universal ODEs because symplectic methods are normally created for systems with a Hamiltonian formulation. There are also alternative symplectic approaches that preserve the system's symplectic structure but use somewhat different update equations, such as the Verlet method. This is explained in [92].

There are two class of numerical integrators to solve rigid body motions: implicit and explicit. In the implicit type of integration techniques outer correctors are involved with either Newmark-β or Crank-Nicolson schemes. Explicit integrators, like the symplectic scheme, are called only once per time step and do not involve outer correctors.

Annexure-III: Section-4.3.3

fvSolution: The fvSolution used in the problem stated in 5.3 is shown below: (source: www.openfoam.com/documentation/user-guide/6-solving/6.2-numerical-schemes)



```
solvers
{
    Phi
    {
        solver          GAMG;
        smoother        DICGaussSeidel;
        tolerance       1e-6;
        relTol          0.01;
    }
    p
    {
        solver          GAMG;
        smoother        DICGaussSeidel;
        tolerance       1e-6;
        relTol          0.01;
    }
    "pcorr.*"
    {
        solver          GAMG;
        tolerance       0.02;
        relTol          0;
        smoother        GaussSeidel;
    }

    pFinal
    {
        solver          GAMG;
        smoother        DICGaussSeidel;
        tolerance       1e-8;
        relTol          0;
    }
```

```
    U
    {
        solver          smoothSolver;
        smoother        GaussSeidel;
        tolerance       1e-06;
        relTol          0;
    }
    UFinal
    {
        solver          smoothSolver;
        smoother        GaussSeidel;
        tolerance       1e-08;
        relTol          0;
    }
    k
    {
        solver          smoothSolver;
        smoother        GaussSeidel;
        tolerance       1e-8;
        relTol          0.1;
        nSweeps         1;
    }
    kFinal
    {
        solver          smoothSolver;
        smoother        GaussSeidel;
        tolerance       1e-8;
        relTol          0.1;
        nSweeps         1;
    }
```

```
    omega
    {
        solver          smoothSolver;
        smoother        GaussSeidel;
        tolerance       1e-8;
        relTol          0.1;
        nSweeps         1;
    }
    omegaFinal
    {
        solver          smoothSolver;
        smoother        GaussSeidel;
        tolerance       1e-8;
        relTol          0.1;
        nSweeps         1;
    }
    epsilon
    {
        solver          smoothSolver;
        smoother        GaussSeidel;
        tolerance       1e-8;
        relTol          0.1;
        nSweeps         1;
    }
    epsilonFinal
    {
        solver          smoothSolver;
        smoother        GaussSeidel;
        tolerance       1e-8;
        relTol          0.1;
        nSweeps         1;
    }
}
```

```
PIMPLE
{
    momentumPredictor   yes;
    nOuterCorrectors    1;
    nCorrectors         1;
    nNonOrthogonalCorrectors 3;
    residualControl
    {
        U
        {
            tolerance  1e-5;
            relTol      0;
        }
        p
        {
            tolerance  5e-4;
            relTol      0;
        }
    }
}

relaxationFactors
{
    fields
    {
        p      1;
        pFinal   1;
    }
    equations
    {
        "U|k|epsilon|omega"     1;
        "(U|k|epsilon|omega)Final"   1;
    }
}
```

Fig. 4.3.1.1: fvSchemes in OpenFOAM

Solvers, relaxationFactors, and PISO, SIMPLE, or PIMPLE are some of the subdictionaries that make up fvSolution and are explained in the following paragraphs.

4.3.1.1) Linear solver control:

Solver is the first sub-dictionary in the fvSolution (Fig. 4.3.1.1) used in the problem stated in Section-5.3. The term "solver" here refers to the method of calculation, to solve a matrix equation. It specifies each linear-solver that is used for each discretized equation. Each item in solvers begins with a keyword that corresponds to the variable being solved for in the particular equation. For instance, the entries for U and p come from the fact that pimpleFoam solves equations for velocity U and pressure p. The term refers to a sub-dictionary that lists the type of solver and the solver's operating settings. The solver keyword allows to choose the solver. The solver types used in this work is given below.

- smoothSolver: solver that uses a smoother.
- GAMG: generalised geometric-algebraic multi-grid.

Asymmetric and symmetric matrices are distinguished by the solvers. The coefficients of a symmetric matrix are formed by temporal derivatives and Laplacian terms, but assymetry is introduced by an advective derivative. The symmetry of the matrix is dependent on the components of the equation being solved. The next sections include descriptions of the parameters, such as tolerance, relTol, preconditioner, etc.

Solution tolerances

The matrices are sparse, which means that their separated, decoupled, finite volume numerics primarily contain coefficients of 0. As a result, the solvers are typically iterative, that is, they are based on decreasing the equation residual over iteratively solving the problem. The residual can be thought of as a measurement of the inaccuracy in the solution; so it is tried to be reduced to increase accuracy of the solution. In order to evaluate the residual more precisely, the current solution is substituted into the equation, and the size of the difference between the left and right sides is taken. The residual is then normalised to make it independent of the size of the problem being studied.

The initial residual is assessed using the field's most recent values prior to solving an equation for that field. The residual is reevaluated following each iteration of the solution. Any one of the following circumstances causes the solver to stop:

- the residue is less than the solver's tolerance;
- the current to initial residuals ratio is less than relTol, the solver's relative tolerance;
- the number of iterations is greater than the specified maximum, maxIter;

The solver tolerance should be used to represent the threshold at which the residual is sufficiently low for the solution to be considered accurate. The solver relative tolerance limits the relative improvement from the initial to the final solution. During transient simulations, the solver relative tolerance is frequently set to 0, forcing the solution to converge to the solver tolerance at each time step. For all solvers, the tolerances, tolerance, and relTol must be defined in the dictionaries; the optional maxIter has a default value of 1000. In a single time step or solution step, equations are frequently solved many times. For instance, the PISO algorithm solves pressure equations in accordance with the number of nCorrectors, as explained in section 3.3. while this occurs, the solver is typically set up to use different settings while resolving the specific problem. The term "Final" is added to the field name in the keyword to accomplish this.

The first two solutions would use the settings for p with relTol of 0.01, so that the cost of calculating each equation is quite minimal. If the case is specified to solve pressure four times within one time step, then only after solving the equation four times does it reach the tolerance-specified residual level (because relTol is 0, thereby deactivating it) for more accuracy but at a higher cost.

Preconditioned conjugate gradient solvers

The preconditioner keyword in the solver dictionary is used to describe a variety of alternatives for preconditioning matrices in conjugate gradient solvers, the ones used are described below. The OpenFOAM tutorials are the only place where the DIC/DILU preconditioners are specified.

- GAMG: geometric-algebraic multi-grid.
- none: no preconditioning.

Smooth solvers

The choice of smoother must be provided for solvers that employ one. The options that are smoother are listed below. In the lessons, the symGaussSeidel and GaussSeidel smoothers are preferred. The number of sweeps before the residual is regenerated when using the smooth solvers can be optionally specified by the nSweeps keyword. If it isn't changed, it defaults to a value of 1.

- GaussSeidel: Gauss-Seidel.
- DICGaussSeidel: diagonal incomplete-Cholesky/LU with Gauss-Seidel (symmetric/asymmetric). The number of sweeps before the residual is regenerated when using the smooth solvers can be optionally specified by the nSweeps keyword. If it isn't changed, it defaults to a value of 1.

Geometric-algebraic multi-grid solvers

In order to find an accurate solution on a fine mesh, the generalised method of geometric-algebraic multi-grid (GAMG) first maps a quick solution onto a mesh with fewer cells. GAMG is quicker than conventional methods when the advantages of solving first on coarser meshes outweigh the extra costs of mesh refinement and field data mapping. Actually, GAMG starts with the user-specified mesh and gradually finenes/coarses it. The user is only required to submit an estimate of the mesh size in terms of the number of cells at the most basic level. The process outlined by the agglomerator keyword is used to aggregate cells. There are various optional entries, the most of which default in the tutorials, can be used to regulate the agglomeration.

The following optional inputs specify the number of sweeps the smoother uses at various mesh densities.

- nPreSweeps: number of sweeps as the algorithm is coarsening (default 0).
- preSweepsLevelMultiplier: multiplier for the the number of sweeps between each coarsening level (default 1).
- maxPreSweeps: maximum number of sweeps as the algorithm is coarsening (default 4).
- nPostSweeps: number of sweeps as the algorithm is refining (default 2).
- postSweepsLevelMultiplier: multiplier for the the number of sweeps between each refinement level (default 1).
- maxPostSweeps: maximum number of sweeps as the algorithm is refining (default 4).
- nFinestSweeps: number of sweeps at finest level (default 2).

Solution under-relaxation

When addressing steady-state problems, in particular, RelaxationFactors, a second sub-dictionary of fvSolution that is often utilised in OpenFOAM, handles under-relaxation, a technique for improving computation stability. Under-relaxation lowers the amount that a variable varies from one iteration to the next by either altering the field directly or by modifying the source and solution matrices before solving for a field. The degree of under-relaxation is determined by an under-relaxation factor, which is defined as, α

- No specified α no under-relaxation.
- $\alpha = 1$: guaranteed matrix diagonal equality/dominance.
- α decreases, under-relaxation increases.
- $\alpha = 0$: solution does not change with successive iterations.

An ideal value for α one that is small enough to guarantee stable computation but large enough to advance the iterative process quickly. In some cases, α values as high as 0.9 can guarantee stability, while values much lower than, say, 0.2, are prohibitively restrictive in terms of slowing down the iterative process.

The relaxation factors for under-relaxing fields are contained in a field sub-dictionary, while the relaxation factors for under-relaxing equations are contained in an equations sub-dictionary.. This fvSolution solves the transient incompressible solver pimpleFoam which simply relies on under-relaxation to guarantee matrix diagonal equality, as is typical of transient simulations.

Pressure referencing

Pressure is relative in a closed, incompressible system; the range of pressures, not the absolute numbers, matters. In these circumstances, the solver assigns pRefValue as the reference level in cell pRefCell. The solvers that need them employ these entries, which are often kept in the SIMPLE, PISO, or PIMPLE sub-dictionaries, depending on the circumstance.

fvSchemes

Schemes used in fvSchemes of the present work is shown in Fig. 4.3.2.1. (source: www.openfoam.com/documentation/user-guide/6-solving/6.3-solution-and-algorithm-control)

```
ddtSchemes
{
    default         Euler;
}
gradSchemes
{
    default         Gauss linear;
    grad(U)         cellLimited Gauss linear 1;
}
divSchemes
{
    default none;
    div(phi,U)        bounded Gauss linearUpwindV grad(U);
    div(phi,k)        bounded Gauss upwind;
    div(phi,epsilon)  bounded Gauss upwind;
    div(phi,omega)  bounded Gauss upwind;
    div((nuEff*dev2(T(grad(U))))) Gauss linear;
}
laplacianSchemes
{
    default         Gauss linear limited corrected 0.5;
}
interpolationSchemes
{
    default         linear;
}
snGradSchemes
{
    default         corrected;
}
wallDist
{
        method          meshWave;
}
```

Fig. 4.3.2.1: fvSchemes

The illustration demonstrates the contents of the fvSchemes dictionarySchemes sub-dictionaries that contain keyword entries for each term specified throughout comprise a default entry as well as additional entries whose names correlate to a word identifier for the particular term specified, *e.g.*grad(p) for ∇p If a default scheme is specified in a particular *...Schemes* sub-dictionary, it is assigned to all of the terms to which the sub-dictionary refers, *e.g.* specifying a default in *gradSchemes* sets the scheme for all gradient terms in the application, *e.g.* ∇p, ∇U. When a default is specified, it is not necessary to specify each specific term itself in that sub-dictionary, *i.e.* the entries for grad(p), grad(U) in this example. However, if any of these terms are included, the specified scheme overrides the default scheme for that term.

As with the divSchemes in the Fig. 4.3.2.1 , the 'none' entry signifies that there is no default scheme specified. In this case, each term in the sub-dictionary is to be explicitly provided.

```
default         backward;
default         CrankNicolson 0.9;
default         Euler;
default         localEuler;
default         none;
default         steadyState;
```

Time Schemes: The ddtSchemes sub-dictionary contains a definition for the first time for derivative terms ($\partial/\partial t$). The discretization schemes are:

- steadyState: sets time derivatives to zero.
- Euler: transient, first order implicit, bounded.
- backward: transient, second order implicit, potentially unbounded.
- CrankNicolson: transient, second order implicit, bounded; requires an off-centering coefficient ψ where:

$$\psi = \begin{cases} 1 & \text{corresponds to pure CrankNicolson} \\ 0 & \text{corresponds to Euler} \end{cases}$$

generally $\psi = 0.9$ is used to bound/stabilise the scheme for practical engineering problems.

- localEuler: pseudo transient for accelerating a solution to steady-state using local-time stepping; first order implicit.

Solvers are typically set up to replicate either steady-state or transient conditions. Since the essential nature of the solver is unaffected by changing the time scheme from steady-state to transient or vice versa, the attempt to solve the problem results in an absurd result. Any second time derivative ($\frac{\partial^2}{\partial t^2}$) terms are specified in the *d2dt2Schemes* sub-dictionary. Only the Euler scheme is available for *d2dt2Schemes*.

Gradient Schemes: The sub-dictionary gradSchemes contains gradient terms. The standard discretization scheme, which is mainly used to gradient terms, is:

```
default         Gauss linear;
```

The Gauss entry describes the typical discretization of Gaussian integration using finite volumes, which calls for interpolating values between cell and face centres. The linear entry, which denotes linear interpolation or central differencing, then provides the interpolation technique.

In some tutorial cases, especially those involving meshes of lower quality, the discretization of particular gradient terms is disregarded in order to increase boundedness and stability. The velocity gradient is the term that is superseded in those circumstances.

```
grad(U)         cellLimited Gauss linear 1;
```

and, less frequently, the gradient of turbulence fields, *e.g.*

```
grad(k)         cellLimited Gauss linear 1;
grad(epsilon)   cellLimited Gauss linear 1;
```

They employ the cellLimited approach, which restricts the gradient so that the face values do not exceed the range of values in neighbouring cells when cell values are extended to faces using the computed gradient. Following the underlying approach, for which 1 ensures boundedness and 0 has no limiting effect, a limiting coefficient is supplied; 1 is always utilised.

Divergence Schemes: The *divSchemes* sub-dictionary contains divergence terms, *i.e.* terms of the form ∇, excluding Laplacian terms (of the form $\nabla.(\Gamma\nabla ...)$). This includes both advection terms, *e.g.* $\nabla.(Uk)$, where velocity U provides the advective flux, and other terms, that are often diffusive in nature, *e.g.* $\nabla.\nu(\nabla U)^T$. The default scheme in divSchemes is typically set to none since terms that are fundamentally dissimilar are contained in a single sub-dictionary. Following that, the non-advective terms typically employ the Gauss integration with linear interpolation, for example

```
div(U)          Gauss linear;
```

The options are wider since one of the main difficulties in CFD numeric is the treatment of advective terms. The advective terms' keyword identifiers typically take the form div(phi), where phi refers to the mass flux for compressible flows and the (volumetric) flux of velocity on the cell faces for constant-density flows, for example, div(phi,U) for the advection of velocity, div(phi,e) for the advection of internal energy, and div(phi,k) for turbulent kinetic energy, etc. Each interpolation technique, including linear, linearUpwind, etc., applies the flux phi and the Gauss integration method to interpolate the advected field to the cell faces. The discretisation has a bounded variant. 'V'-schemes (with keywords ending "V"), and rarely-used schemes such as Gauss cubic and vanLeerV, the interpolation schemes used in the tutorials are as follows.

- linear: second order, unbounded.
- linearUpwind: second order, upwind-biased, unbounded, that requires discretization of the velocity gradient to be specified.
- upwind: first-order bounded, generally too inaccurate to be recommended.

- limitedLinear: linear scheme that limits towards upwind in regions of rapidly changing gradient; requires a coefficient, where 1 is strongest limiting, tending towards linear as the coefficient tends to 0.

- LUST: blended 75% linear/ 25%linearUpwind scheme, that requires discretisation of the velocity gradient to be specified.

```
div(phi,U)      Gauss linearUpwind grad(U);
div(phi,U)      Gauss upwind;
```

Specialised variations of schemes created for vector fields are called "V"-schemes. Instead of calculating separate limiters for each component of the vector, they calculate a single limiter that is applied to all of the components of the vectors, which is how they differ from typical systems. The single limiter of the "V"-schemes is determined based on the gradient's direction that changes the most quickly, yielding the strongest limiter that is also the most stable but perhaps less precise.

```
div(phi,U)      Gauss limitedLinearV 1;
div(phi,U)      Gauss linearUpwindV grad(U);
```

The treatment of the material time derivative, which can be described in terms of a spatial time derivative and convection, such as for field e in incompressible flow, is addressed by the bounded variants of schemes. For field e in incompressible flow

$$\frac{De}{Dt} = \frac{\partial e}{\partial t} + U \cdot \nabla e = \frac{\partial e}{\partial t} + \nabla.(Ue) - (\nabla.U)e$$

For numerical solution of incompressible flows, $\nabla.U = 0$ at convergence, at which point the third term on the right hand side is zero. Before convergence is reached, however, $\nabla.U \neq 0$ and in some circumstances, particularly steady-state simulations, it is better to include the third term within a numerical solution because it helps maintain boundedness of the solution variable and promotes better convergence.

For incompressible flow numerical solution, delU at convergence, where the third term on the right hand side is zero. However, delU and in some cases, especially in steady-state simulations, it is preferable to incorporate the third component into a numerical solution since it helps maintain the boundedness of the solution variable and encourages better convergence. This is provided by the bounded variation of the Gauss scheme, which includes the discretization of the third-term with the advection term by default. The following example syntax is used in fvSchemes files for steady-state instances, such as in the simpleFoam tutorials.

```
div(phi,U)      bounded Gauss limitedLinearV 1;
div(phi,U)      bounded Gauss linearUpwindV grad(U);
```

The schemes used for advection of scalar fields are alike to those for advection of velocity, albeit in most cases boundedness is prioritised over accuracy when choosing the advection schemes. For instance, the following is revealed when looking for internal energy advection schemes (e).

foamSearch $FOAM_TUTORIALS fvSchemes "divSchemes.div(phi,e)"

```
div(phi,e)      bounded Gauss upwind;
div(phi,e)      Gauss limitedLinear 1;
div(phi,e)      Gauss LUST grad(e);
div(phi,e)      Gauss upwind;
div(phi,e)      Gauss vanLeer;
```

There are no predefined scenarios that would use linear or linearUpwind velocity instead of advection. Instead, upwind and limitedLinear are widely used, with vanLeer—another limited scheme—appearing at times and having weaker limiting than limitedLinear. There are specialised variations of the limited approaches for scalar fields that are usually bound between 0 and 1. Finding the discretization for advection in the equation for laminar flame transfer leads to:

```
div(phiSt,b)    Gauss limitedLinear01 1;
```

By adding 01 to the name of the scheme, the underlying scheme, limitedLinear, is specifically tailored for stronger bounding between 0 and 1.

Surface normal gradient schemes: The *snGradSchemes* sub-dictionary contains surface normal gradient words, necessary to evaluate a Laplacian term using Gaussian integration. A cell face is where a surface normal gradient is evaluated. It is the component of the gradient of values at the centres of the two cells that the face connects that is normal to the face. ‘default’ scheme for snGradSchemes are:

```
 default        corrected;
default         limited corrected 0.33;
default         limited corrected 0.5;
default         orthogonal;
default         uncorrected;
```

The formula for calculating the gradient at a face starts by dividing the distance between the value at the centre of the cell on one side of the face and the value at the centre on the other. If the vector connecting the cell centres is orthogonal to the face, that is, if they are at right angles, the calculation is second-order accurate for the gradient normal to the face. The orthogonal system is as follows. Meshes for real-world engineering geometries generally lack the regular meshes necessary for orthogonality, which are typically aligned with the Catersian coordinate system.

Therefore, an explicit non-orthogonal correction can be applied to the orthogonal component, known as the corrected scheme, to retain second-order precision. As the non-orthonality, or angle alpha between the face normal vector and cell-cell vector, grows, so does the magnitude of the correction.

As α tends towards 90^0, *e.g.,* beyond 70^0, the explicit correction can be so large to cause a solution to go unstable. The solution can be stabilised by applying the limited scheme to the correction which requires a coefficient ψ, $0 \leq \psi \leq 1$ where,

$$\psi = \begin{cases} 0 & \text{corresponds to uncorrected} \\ 0.333 & \text{non} - \text{orthogonal correction} \leq 0.5 \times \text{orthogonal} \\ 0.5 & \text{non} - \text{orthogonal correction} \leq \text{orthogonal part} \\ 1 & \text{corresponds to corrected} \end{cases}$$

Typically, ψ is chosen to be 0.33 or 0.5, where 0.33 offers greater stability and 0.5 greater accuracy.

The corrected scheme applies under-relaxation in which the implicit orthogonal calculation is increased by $\cos^{-1}\alpha$, with an equivalent boost within the non-orthogonal correction. The uncorrected scheme is equivalent to the corrected scheme, without the non-orthogonal correction, so includes is like orthogonal but with the $\cos^{-1}\alpha$ under-relaxation.

The uncorrected and orthogonal schemes are often only advised for meshes with extremely minimal non-orthogonality (e.g., no more than 5°). Although the corrected method is typically advised, but 'limited' may be necessary for maximal non-orthogonality above 70°. Convergence is typically difficult to attain at non-orthogonality above 80°.

Laplacian schemes: The *laplacianSchemes* sub-dictionary contains Laplacian terms. A typical Laplacian term is $\nabla.(\nu\nabla U)$, the diffusion term in the momentum equations, which corresponds to the keyword laplacian(ν,U) in *laplacianSchemes*. The *Gauss* scheme is the only choice of discretisation and requires a selection of both an interpolation scheme for the diffusion coefficient, i.e. ν, and a surface normal gradient scheme, i.e. ∇U. To summarise, the entries required are:

Gauss <interpolationScheme> <snGradScheme>

It reveals the following entries.

default Gauss linear corrected;
default Gauss linear limited corrected 0.33;
default Gauss linear limited corrected 0.5;

default Gauss linear orthogonal;
default Gauss linear uncorrected;

In all cases, the linear interpolation scheme is used for interpolation of the diffusivity. The cases use the same array of *snGradSchemes* based on level on non-orthogonality, as described in Annexure-I: Section-3.3.1.

Interpolation schemes: The terms in the *interpolationSchemes* sub-dictionary are interpolations of values, usually from cell centres to face centres, and are largely applied to the interpolation of velocity to face centres for the computation of flux phi. Although OpenFOAM has a variety of interpolation methods, a search for the default scheme in all the tutorial cases indicates that linear interpolation is used in nearly all instances, with the exception of a few rare situations, such as DNS on a regular mesh and stress analysis, where cubic interpolation is employed.

ANNEXURE-III: RESULTS

Annexure-III: Section-5.3

The additional information on snappyHexMesh mentioned in Section-5.3.2.1.2 are detailed in this section.

Annexure-III: Section -5.3.2.1.2

The addlayers commands of snappyHexMesh to add extra layers of cells near the wall of the structure to capture better results is shown in the following figures:

```
// Settings for the layer addition.
addLayersControls
{
    // Are the thickness parameters below relative to the undistorted
    // size of the refined cell outside layer (true) or absolute sizes (false).
    relativeSizes true;

    // Layer thickness specification. This can be specified in one of following
    // ways:
    // - expansionRatio and finalLayerThickness (cell nearest internal mesh)
    // - expansionRatio and firstLayerThickness (cell on surface)
    // - overall thickness and firstLayerThickness
    // - overall thickness and finalLayerThickness
    // - overall thickness and expansionRatio
    //
    // Note: the mode thus selected is global, i.e. one cannot override the
    //       mode on a per-patch basis (only the values can be overridden)

        // Expansion factor for layer mesh
        //expansionRatio 1.0;
        expansionRatio 1.2;
```

```
    minThickness 0.1;

    // Per final patch (so not geometry!) the layer information
    // Note: This behaviour changed after 21x. Any non-mentioned patches
    //       now slide unless:
    //          - nSurfaceLayers is explicitly mentioned to be 0.
    //          - angle to nearest surface < slipFeatureAngle (see below)
    layers
    {
        cylinder
        {
            nSurfaceLayers 5;   //10
        }
```

```
        // Stop layer growth on highly warped cells
        maxFaceThicknessRatio 0.5;

    // Patch displacement

        // Number of smoothing iterations of surface normals
        nSmoothSurfaceNormals 1;

        // Smooth layer thickness over surface patches
        nSmoothThickness 10;
```

```
    // Medial axis analysis

        // Angle used to pick up medial axis points
        // Note: changed(corrected) w.r.t 17x! 90 degrees corresponds to 130
        // in 17x.
        minMedialAxisAngle 90;
        minMedianAxisAngle 90;

        // Reduce layer growth where ratio thickness to medial
        // distance is large
        maxThicknessToMedialRatio 0.3;

        // Number of smoothing iterations of interior mesh movement direction
        nSmoothNormals 3;
```

```
        nRelaxIter 5;

        // Create buffer region for new layer terminations
        nBufferCellsNoExtrude 0;

        // Overall max number of layer addition iterations. The mesher will
        // exit if it reaches this number of iterations; possibly with an
        // illegal mesh.
        nLayerIter 50;

        // Max number of iterations after which relaxed meshQuality controls
        // get used. Up to nRelaxedIter it uses the settings in
        // meshQualityControls,
        // after nRelaxedIter it uses the values in
        // meshQualityControls::relaxed.
        nRelaxedIter 20;

        // Additional reporting: if there are just a few faces where there
        // are mesh errors (after adding the layers) print their face centres.
        // This helps in tracking down problematic mesh areas.
        //additionalReporting true;
}
```

```
    relaxed
    {
        // Maximum non-orthogonality allowed. Set to 180 to disable.
        maxNonOrtho 75;
    }
    //minFlatness 0.5;
    // Advanced
        // Number of error distribution iterations
        nSmoothScale 4;
        // amount to scale back displacement at error points
        errorReduction 0.75;
}
```

Fig. 5.9: Commands for adding layers near the structure wall

Annexure-III: Section -5.4

[These results are a part of a published work in 9th International and 49th National Conference on Fluid Mechanics and Fluid Power 2022]

5.4.1 Validation 3: TLD-structure interaction

In order to validate the developed TLD-structure interaction model, the shaking table experiment performed by [93] is considered. TLD tank is 59cm long, and 33.5cm wide. The water depth is 3cm. The frequency ratio is the ratio between the natural frequency of the structure to that of the TLD liquid (here water). The structural displacement with and without attached TLD for different frequency ratios is presented in Fig. 5.4.1. The frequency ratio vs maximum and minimum wave height is presented in Fig. 5.4.2.

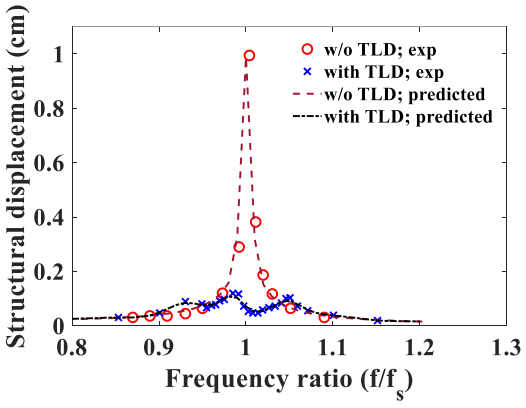


Fig 5.4.1. Structural displacement with and without TLD [Sun et al., 1992]

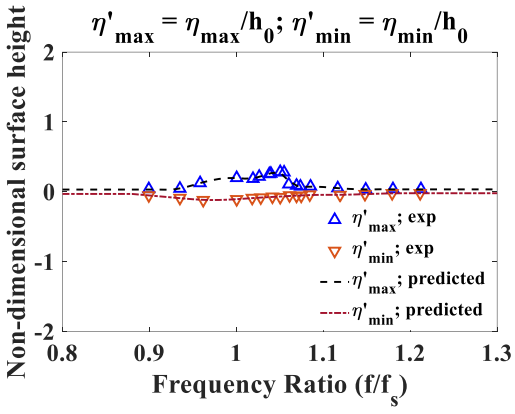


Fig 5.4.2. Maximum and minimum surface height [Sun et al., 1992]

Once the turbulence modelling around a bluff body and the TLD-structure interaction modelling are validated, next two case studies are performed for laminar (Re = 250) and turbulent (Re = 10^5) flows. In both the cases time-varying drag force is used as the forcing to the next stage of TLD-structure interaction model. The general problem schematic is presented earlier in Fig. 5.4.1. The results are presented and discussed in sections 1.1 and 1.2.

1.1 Turbulence-structure-TLD interaction; Re = 250

In this segment, a laminar flow (Re = 250) is considered. The computed drag coefficient and Strouhal numbers are presented in Table 5.4.2. This drag coefficient is subsequently used to estimate the force and the TLD parameters are so chosen that it is tuned to the corresponding natural frequency of 0.538Hz (for St = 0.138). The predicted damped and undamped displacement (non-dimensionalized by the cylinder dimension, D) is presented in Fig. 5.4.2. The inherent structural damping is taken as 0.32% [Sun et al., 1992]. The spring stiffness is calculated accordingly.

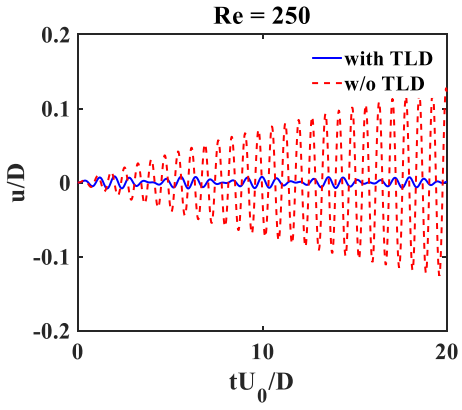


Fig. 5.4.3. Structural response; natural frequency of 0.538Hz;

Table 5.4.2: Drag coefficient for Re = 250 flow case

	Re	St	C_D
Numerical (Franke 1990) [13]	250	0.141	1.67
Present study	250	0.138	1.78

1.2 Turbulence-structure-TLD interaction; Re = 10^5

In this segment, a turbulent flow (Re = 10^5) is considered. The computed drag coefficient is already presented in Table 5.4.1. This drag coefficient is subsequently used to estimate the force.

As the turbulent flow produces a broadband spectrum it is very difficult to find any particular forcing frequency, and eventually tuning the TLD also becomes challenging. In order to estimate the TLD performance in the reduction of the structural response, an extensive study is carried out in two stages.

a) TLD properties are varied to obtain different natural frequencies, replicating a broadband spectrum.

b) For each frequency the TLD-structure interaction is simulated and the performance is measured in terms of the ratio of the maximum displacement with and without TLD.

The result of the performance study is presented in Fig. 5.4.4.

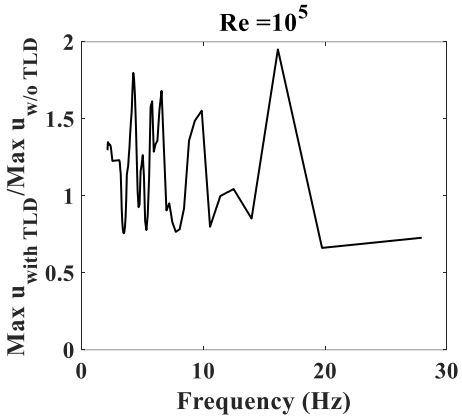


Fig 5.4.4. Performance of TLD in a broadband spectrum

From this study, few frequencies are identified where the TLD can reduce the structural response. In the other frequencies, the effect of attached TLD is found to be adverse though. Next, the structural response with and without TLD at a few identified frequencies are presented in Fig. 5.4.5 and Fig. 5.4.6.

The predicted damped and undamped displacements are non-dimensionalized by the cylinder dimension, D.

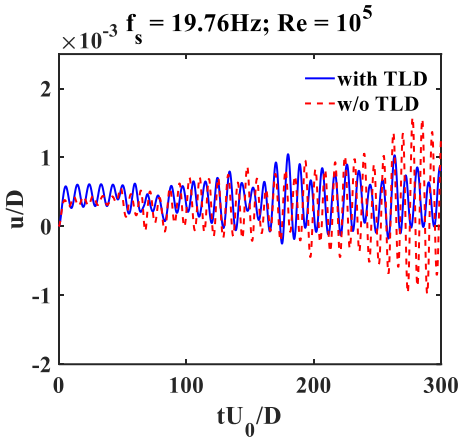


Fig 5.4.5. TLD performance at 19.76Hz frequency

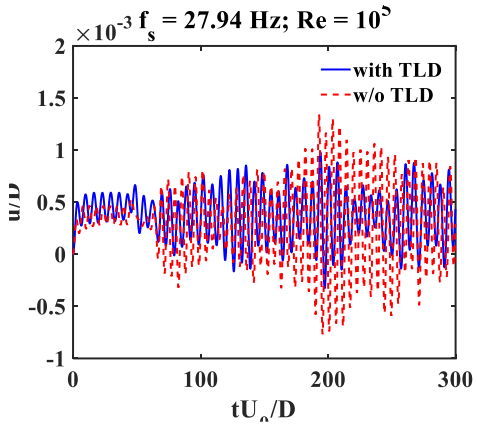


Fig 5.4.6. TLD performance at 27.94Hz frequency

Conclusions: In the present research work, a turbulence-structure TLD model is proposed, developed in FD framework. Open-source flow solver (OpenFOAM) and in-house MATLAB based TLD- structure interaction model is used. The significant conclusions are:

a) In case of laminar flow induced structural vibration, the TLD works perfectly when it is tuned with the vortex-shedding frequency.

b) In case of turbulence, the broadband force spectrum restricts to select any particular tuning frequency. However, in the present work a detailed study is carried out to identify few possible tuning frequencies that eventually is found to be reducing the structural response for the particular present case.

c) There is a requirement of further extensive studies with different turbulent conditions involving two-way FSI model to reach at any practical conclusive decision.